

PIC18FXX2

数据手册

带有 10 位 A/D 的高性能

增强型闪存单片机

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 "牢不可破"的。

代码保护功能处于持续发展中。 Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了 《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip 位于美国亚利桑那州Chandler 和Tempe 及位于加利福尼亚州Mountain View 的全球总部、设计中心和晶圆生产厂均于2003 年10 月通过了 ISO/TS-16949:2002 质量体系认证。公司在 PICmicro® 8 位单片机、 KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品方面的质量体系流程均符合 ISO/TS-16949:2002。此外， Microchip 在开发系统的设计和生产方面的质量体系也已通过了ISO 9001:2000 认证。



PIC18FXX2

带 10 位 A/D 的 28/40 引脚
高性能增强型闪存单片机

高性能 RISC CPU：

- 优化的 C 语言编译器架构 / 指令集
  - 源代码与 PIC16 指令集和 PIC17 指令集兼容
- 程序存储器线性寻址达 32 KB
- 数据存储器线性寻址达 1.5 KB

器件	片内程序存储器		片内 RAM （字节）	数据 EEPROM （字节）
	闪存 （字节）	单字指令数		
PIC18F242	16K	8192	768	256
PIC18F252	32K	16384	1536	256
PIC18F442	16K	8192	768	256
PIC18F452	32K	16384	1536	256

- 高达 10 MIPS 的操作 :
  - DC - 40 MHz 振荡器 / 时钟输入
  - 4 MHz - 10 MHz 振荡器 / 带 PLL 的时钟输入
- 16 位宽的指令总线， 8 位宽的数据总线
- 中断优先级
- 8 x 8 单周期硬件乘法器

外设特性：

- 高灌电流 / 拉电流 25 mA/25 mA
- 三个外部中断引脚
- Timer0 模块：带 8 位可编程的预分频器的 8 位 / 16 位定时器 / 计数器
- Timer1 模块：16 位定时器 / 计数器
- Timer2 模块：带 8 位周期寄存器 （作为 PWM 时基）的定时器 / 计数器
- Timer3 模块：16 位定时器 / 计数器
- 副振荡器时钟选项：Timer1/Timer3
- 两个捕捉 / 比较 /PWM （CCP）模块。 CCP 引脚可以配置为：
  - 捕捉输入：捕捉为 16 位，最大分辨率为 6.25 ns （TCY/16）
  - 比较为 16 位，最大分辨率为 100 ns （TCY）
  - PWM 输出：PWM 分辨率为 1 到 10 位，8 位分辨率时最大 PWM 频率为 156 kHz
10 位分辨率时最大 PWM 频率为 39 kHz
- 主同步串行口 （Master Synchronous Serial Port， MSSP）模块，两种操作模式：
  - 3 线 SPI™ （支持所有的 4 种 SPI 模式）
  - I^2C™ 主从模式

外设特点 （续）：

- 可寻址的 USART 模块：
  - 支持 RS-485 和 RS-232
- 并行从动端口 （Parallel Slave Port， PSP）模块

模拟特性：

- 兼容的 10 位模数转换器模块 （A/D），具有：
  - 快速采样率
  - 休眠期间可以进行转换
  - 线性度小于等于 1 LSb
- 可编程低压检测 （Programmable Low Voltage Detection， PLVD）
  - 支持低压检测中断
- 可编程欠压复位 （Brown-out Reset， BOR）

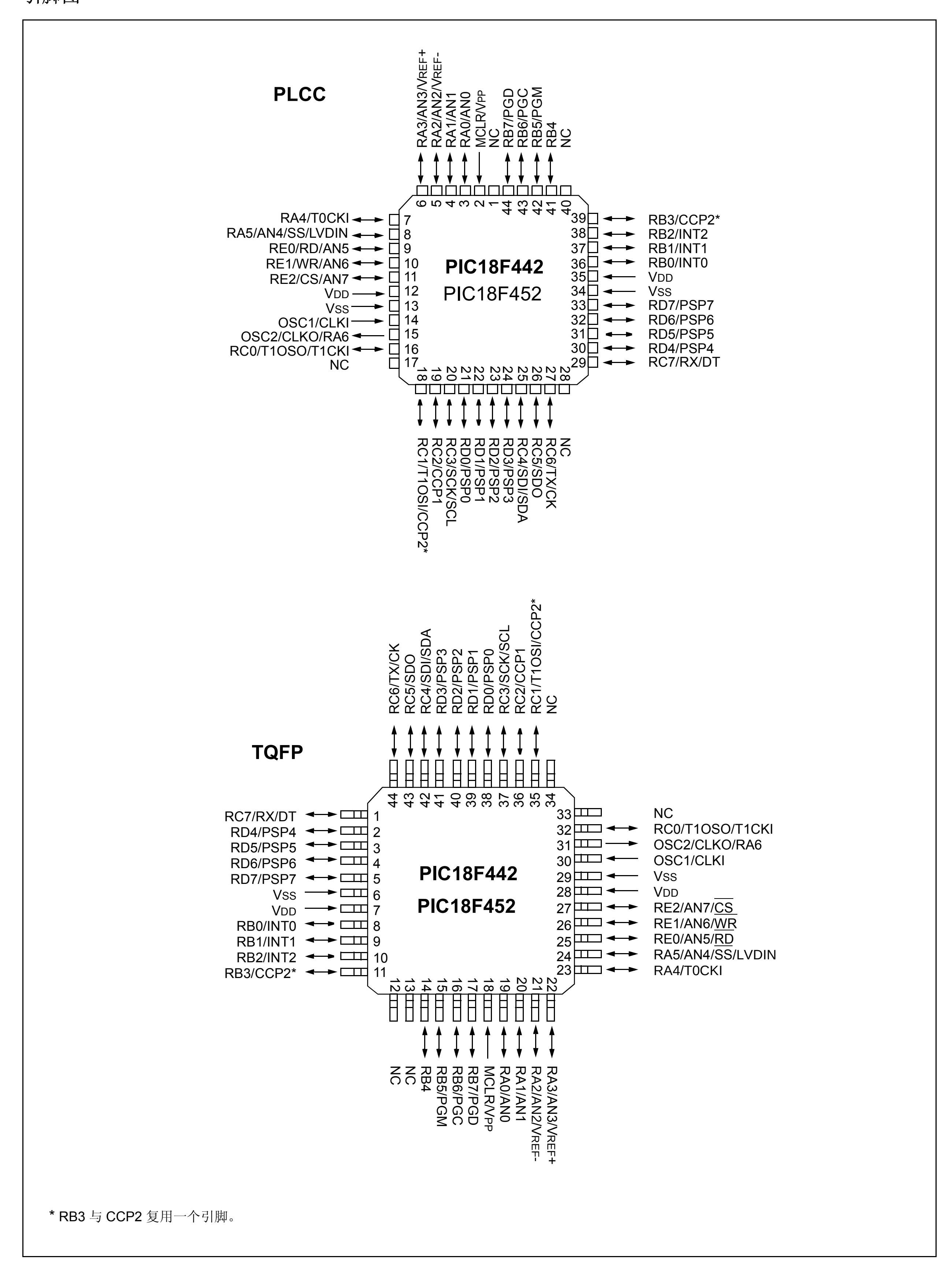
特殊单片机特性：

- 增强型典型闪存程序存储器可擦写 100,000 次
- 数据 EEPROM 存储器可擦写 1,000,000 次
- 闪存 / 数据 EEPROM 保存期大于 40 年
- 软件控制下可自行再编程
- 上电复位 （Power-on Reset， POR）、上电延时定时器 （Power-up Timer， PWRT）和振荡器起振定时器 （Oscillator Start-up Timer， OST）
- 带有独立的片内 RC 振荡器的看门狗定时器 （Watchdog Timer， WDT）可保证运行可靠
- 可编程代码保护
- 省电的休眠模式
- 用户可选的振荡器包括：
  - 4 倍锁相环 （用于主振荡器）
  - 32 kHz 副振荡器时钟输入
- 由 5V 单电源供电，通过两引脚电路进行在线串行编程 （In-Circuit Serial Programming™， ICSP™）
- 通过两引脚进行在线调试 （In-Circuit Debug， ICD）

CMOS 技术：

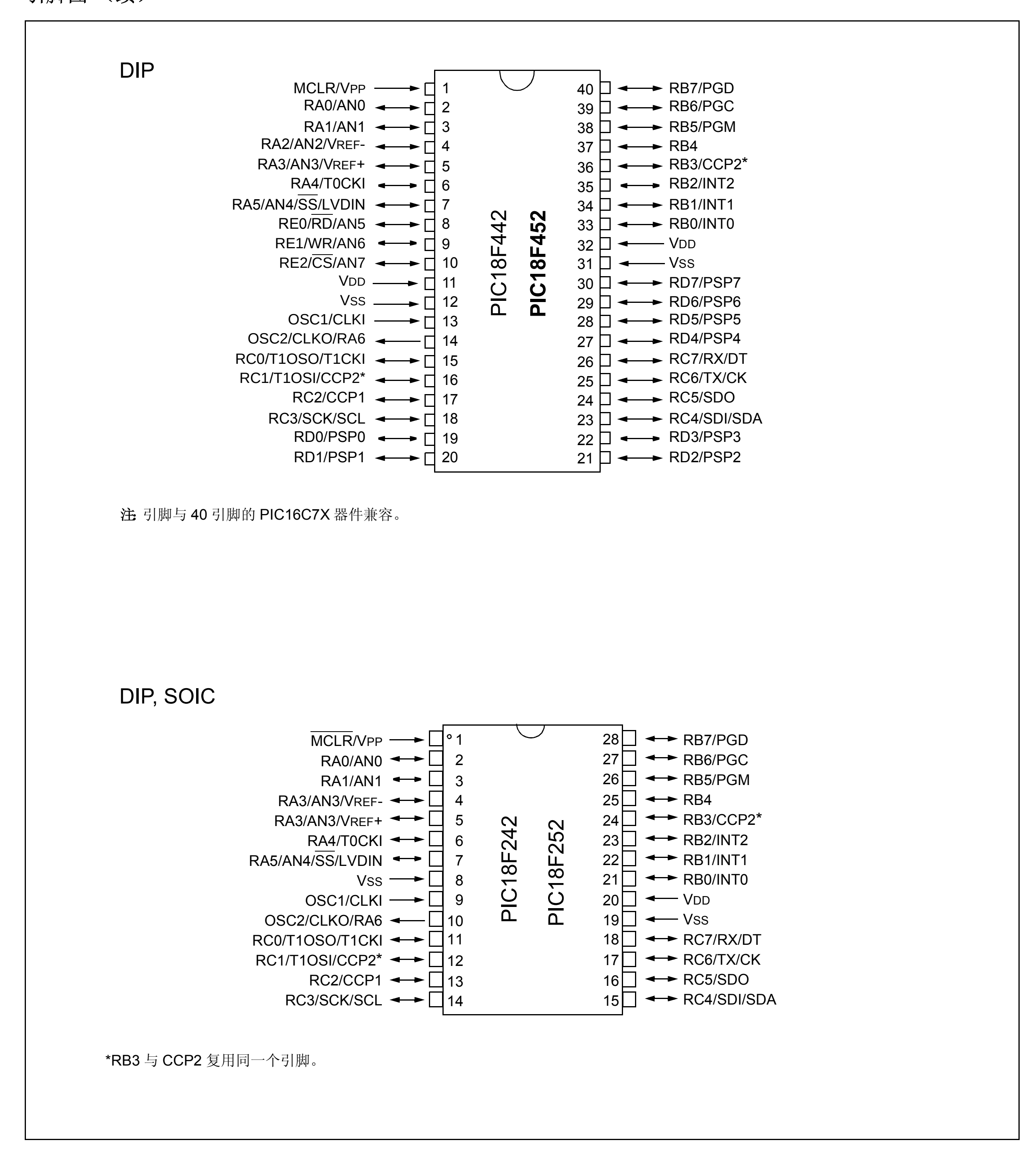
- 低功耗高速闪存 /EEPROM 技术
- 全静态设计
- 宽工作电压范围 （2.0V 到 5.5V）
- 工业级温度范围和扩展级温度范围
- 低功耗：
  - 在 5V， 4 MHz 时，典型值小于 1.6 mA
  - 在 3V， 32 kHz 时，典型值为 25µA
  - 小于 0.2µA 的典型待机电流

引脚图



* RB3 与 CCP2 复用一个引脚。

引脚图 （续）


DIP
MCLR/VPP
RA0/AN0
RA1/AN1
RA2/AN2/VREF-
RA3/AN3/VREF+
RA4/T0CKI
RA5/AN4/SS/LVDIN
RE0/RD/AN5
RE1/WR/AN6
RE2/CS/AN7
VDD
VSS
OSC1/CLKI
OSC2/CLKO/RA6
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2*
RC2/CCP1
RC3/SCK/SCL
RD0/PSP0
RD1/PSP1
PIC18F442
PIC18F452
RB7/PGD
RB6/PGC
RB5/PGM
RB4
RB3/CCP2*
RB2/INT2
RB1/INT1
RB0/INT0
VDD
VSS
RD7/PSP7
RD6/PSP6
RD5/PSP5
RD4/PSP4
RC7/RX/DT
RC6/TX/CK
RC5/SDO
RC4/SDI/SDA
RD3/PSP3
RD2/PSP2
注: 引脚与 40 引脚的 PIC16C7X 器件兼容。
DIP, SOIC
MCLR/VPP
RA0/AN0
RA1/AN1
RA3/AN3/VREF-
RA3/AN3/VREF+
RA4/T0CKI
RA5/AN4/SS/LVDIN
VSS
OSC1/CLKI
OSC2/CLKO/RA6
RC0/T1OSO/T1CKI
RC1/T1OSI/CCP2*
RC2/CCP1
RC3/SCK/SCL
PIC18F242
PIC18F252
RB7/PGD
RB6/PGC
RB5/PGM
RB4
RB3/CCP2*
RB2/INT2
RB1/INT1
RB0/INT0
VDD
VSS
RC7/RX/DT
RC6/TX/CK
RC5/SDO
RC4/SDI/SDA
*RB3 与 CCP2 复用同一个引脚。

目录

致 客 户

我们旨在提供最佳文档供客户正确使用 Microchip 产品。为此，我们将不断改进出版物的内容和质量，使之更好地满足您的要求。出版物的质量将随新文档及更新版本的推出而得到提升。

如果您对本出版物有任何问题和建议，请通过电子邮件联系我公司 TRC 经理，电子邮件地址为 **CTRC@microchip.com**，或将本数据手册后附的 **《读者反馈表》**传真到 86-21-5407 5066。我们期待您的反馈。

最新数据手册

欲获得本数据手册的最新版本，请查询我公司的网站：

http://www.microchip.com

查看数据手册中任意一页下边角处的文献编号即可确定其版本。文献编号中数字串后的字母是版本号，例如：DS30000A是DS30000 的 A 版本。

勘误表

现有器件可能带有一份勘误表，描述了实际运行与数据手册中记载内容之间存在的细微差异以及建议的变通方法。一旦我们了解到器件 / 文档存在某些差异时，就会发布勘误表。勘误表上将注明其所适用的硅片版本和文件版本。

欲了解某一器件是否存在勘误表，请通过以下方式之一查询：

- Microchip 网站 http://www.microchip.com
- 当地 Microchip 销售办事处 （见最后一页）

在联络销售办事处时，请说明您所使用的器件型号、硅片版本和数据手册版本 （包括文献编号）。

客户通知系统

欲及时获知 Microchip 产品的最新信息，请到我公司网站 www.microchip.com 上注册。

注：

1.0 器件概述

本文包含针对以下器件的信息：

- PIC18F242
- PIC18F252
- PIC18F442
- PIC18F452

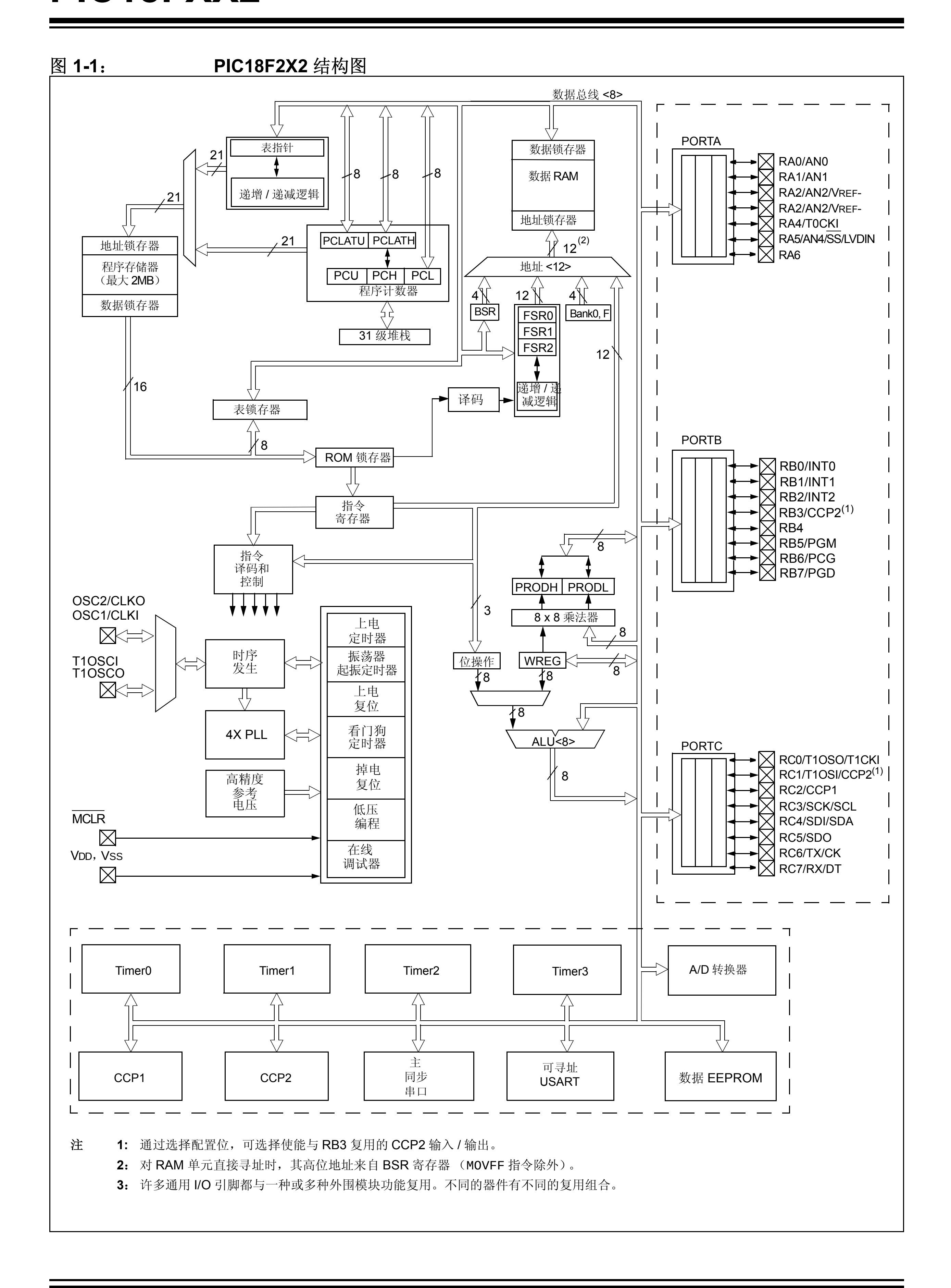
这些器件分为 28 引脚封装和 40/44 引脚封装。28 引脚的器件不带并行从动端口 （PSP），并且模数 （A/D）转换器输入通道的数量也减为5个。表 1-1是特性概述。

后面的两个图是按引脚数分类的器件结构图。对应28引脚器件是图 1-1，对应40/44引脚的是图 1-2。表 1-2和表 1-3 分别列出了 28 引脚器件和 40/44 引脚器件的引脚排列。

表 1-1：　　器件特性

特性	PIC18F242	PIC18F252	PIC18F442	PIC18F452
工作频率	DC - 40 MHz	DC - 40 MHz	DC - 40 MHz	DC - 40 MHz
程序存储器 （字节）	16K	32K	16K	32K
程序存储器 （指令数）	8192	16384	8192	16384
数据存储器 （字节）	768	1536	768	1536
数据 EEPROM 存储器 （字节）	256	256	256	256
中断源	17	17	18	18
I/O 端口	端口 A、 B、 C	端口 A、 B、 C	端口 A、B、C、D、E	端口 A、 B、 C、 D、 E
定时器	4	4	4	4
捕捉 / 比较 /PWM 模块	2	2	2	2
串行通信	MSSP，可寻址 USART	MSSP，可寻址 USART	MSSP，可寻址 USART	MSSP，可寻址 USART
并行通信	—	—	PSP	PSP
10 位模数转换模块	5 个输入通道	5 个输入通道	8 个输入通道	8 个输入通道
复位 （和延时）	POR， BOR，RESET 指令，堆栈满，堆栈下溢（PWRT， OST）	POR， BOR，RESET 指令，堆栈满，堆栈下溢（PWRT， OST）	POR， BOR，RESET 指令，堆栈满，堆栈下溢（PWRT， OST）	POR， BOR，RESET 指令，堆栈满，堆栈下溢（PWRT， OST）
可编程低压检测	有	有	有	有
可编程欠压复位	有	有	有	有
指令集	75 条指令	75 条指令	75 条指令	75 条指令
封装	28 引脚 DIP 28 引脚 SOIC	28 引脚 DIP 28 引脚 SOIC	40 引脚 DIP 44 引脚 PLCC 44 引脚 TQFP	40 引脚 DIP 44 引脚 PLCC 44 引脚 TQFP

图 1-1： PIC18F2X2 结构图

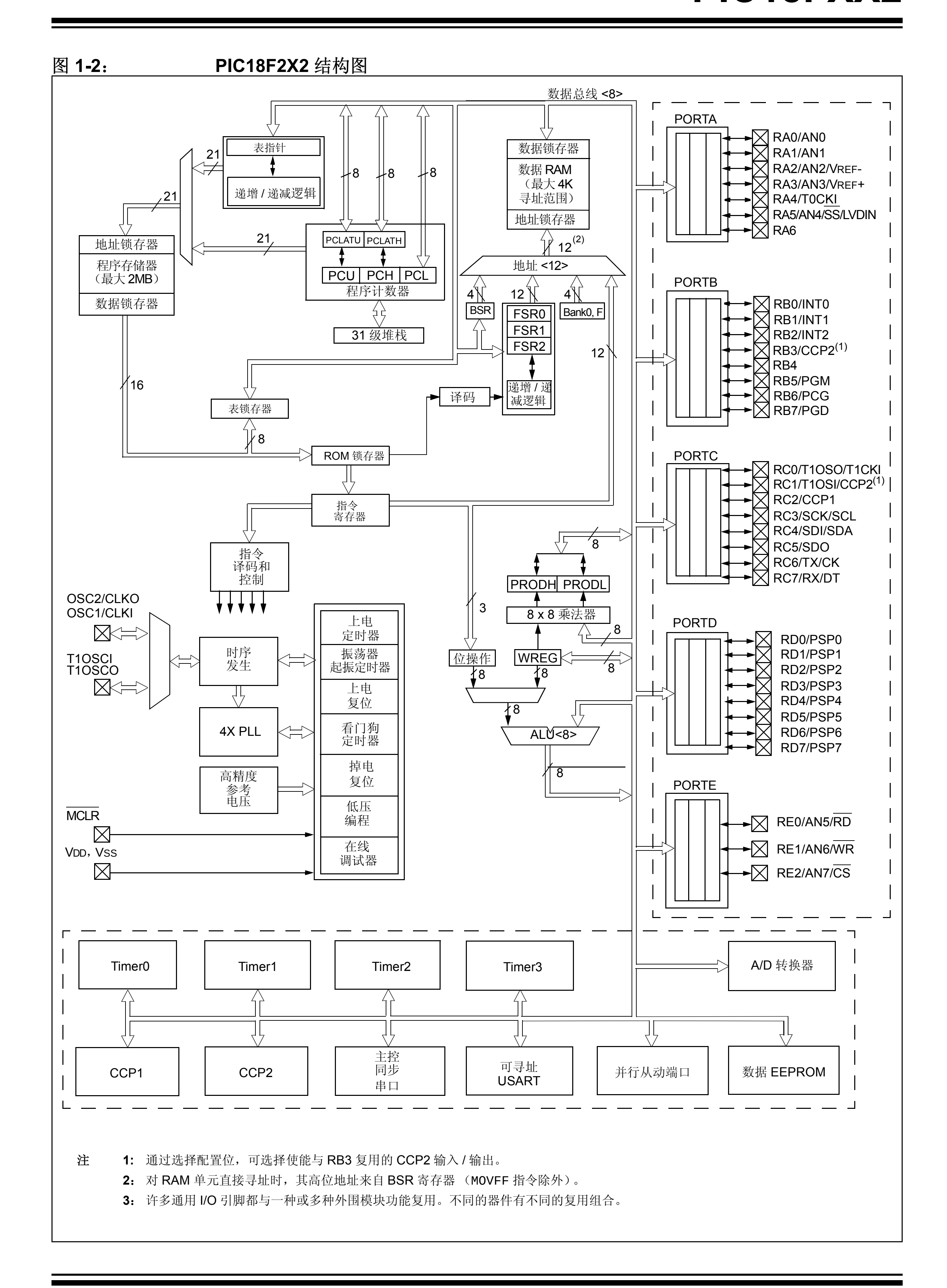


注 1：通过选择配置位，可选择使能与 RB3 复用的 CCP2 输入 / 输出。

2：对 RAM 单元直接寻址时，其高位地址来自 BSR 寄存器 （MOVFF 指令除外）。

3：许多通用 I/O 引脚都与一种或多种外围模块功能复用。不同的器件有不同的复用组合。

图 1-2：　　PIC18F2X2 结构图



注　1：通过选择配置位，可选择使能与 RB3 复用的 CCP2 输入 / 输出。

　　2：对 RAM 单元直接寻址时，其高位地址来自 BSR 寄存器 （`MOVFF` 指令除外）。

　　3：许多通用 I/O 引脚都与一种或多种外围模块功能复用。不同的器件有不同的复用组合。

表 **1-2**： **PIC18F2X2** 引脚排列 **I/O** 说明

引脚名	引脚号		引脚类型	缓冲器类型	说明
	DIP	SOIC			
$\overline{MCLR}$/VPP	1	1			主清零 （输入）或高电平 ICSP 编程使能引脚
MCLR			I	ST	主清零 （复位）输入。此引脚为低电平时，器件复位 （低电平有效）。
VPP			I	ST	高电平 ICSP 编程使能引脚。
NC	—	—	—	—	这些引脚应该悬空。
OSC1/CLKI	9	9			晶振或外部时钟输入。
OSC1			I	ST	晶振输入或外部时钟源输入。配置为 RC 模式时带 ST 缓冲器，否则带 CMOS 缓冲器。
CLKI			I	CMOS	外部时钟源输入。总是与 OSC1 引脚功能复用。（参见有关 OSC1/CLKI 和 OSC2/CLKO 引脚的信息。）
OSC2/CLKO/RA6	10	10			晶振或时钟输出。
OSC2			O	—	晶振输出。在晶振模式时，该引脚与晶振或谐振器相连。
CLKO			O	—	在 RC 模式时， OSC2 引脚输出 CLKO 振荡信号，该信号是 OSC1 引脚上的振荡信号的 4 分频，等于指令周期。
RA6			I/O	TTL	通用 I/O 引脚。
					PORTA 是双向 I/O 口。
RA0/AN0	2	2			
RA0			I/O	TTL	数字 I/O。
AN0			I	模拟	模拟输入 0。
RA1/AN1	3	3			
RA1			I/O	TTL	数字 I/O。
AN1			I	模拟	模拟输入 1。
RA2/AN2/VREF-	4	4			
RA2			I/O	TTL	数字 I/O。
AN2			I	模拟	模拟输入 2。
VREF-			I	模拟	A/D 参考电压 （低）输入。
RA3/AN3/VREF+	5	5			
RA3			I/O	TTL	数字 I/O。
AN3			I	模拟	模拟输入 3。
VREF+			I	模拟	A/D 参考电压 （高）输入。
RA4/T0CKI	6	6			
RA4			I/O	ST/OD	数字 I/O。当配置为输出时，为漏极开路。
T0CKI			I	ST	Timer0 外部时钟输入。
RA5/AN4/$\overline{SS}$/LVDIN	7	7			
RA5			I/O	TTL	数字 I/O。
AN4			I	模拟	模拟输入 4。
$\overline{SS}$			I	ST	SPI 从模式选择输入。
LVDIN			I	模拟	低压检测输入。
RA6					参见 OSC2/CLKO/RA6 引脚。

注： TTL=TTL 兼容输入　　CMOS=CMOS 兼容输入或输出
ST=CMOS 电平的施密特触发器输入　　I= 输入
O= 输出　　P= 电源
OD= 漏极开路 （没有二极管接到 VDD）

表 **1-2**： **PIC18F2X2** 引脚排列 **I/O** 说明 （续）

引脚名	引脚号 DIP	引脚号 SOIC	引脚类型	缓冲器类型	说明
					PORTB 是双向 I/O 口。PORTB 所有输入端都有可编程的内部弱上拉。
RB0/INT0	21	21			
RB0			I/O	TTL	数字 I/O。
INT0			I	ST	外部中断 0。
RB1/INT1	22	22			
RB1			I/O	TTL	数字 I/O。
INT1			I	ST	外部中断 1。
RB2/INT2	23	23			
RB2			I/O	TTL	数字 I/O。
INT2			I	ST	外部中断 2。
RB3/CCP2	24	24			
RB3			I/O	TTL	数字 I/O。
CCP2			I/O	ST	Capture2 输入、 Compare2 输出、 PWM2 输出。
RB4	25	25	I/O	TTL	数字 I/O。 电平变化中断引脚。
RB5/PGM	26	26			
RB5			I/O	TTL	数字 I/O。电平变化中断引脚。
PGM			I/O	ST	低压 ICSP 编程使能引脚。
RB6/PGC	27	27			
RB6			I/O	TTL	数字 I/O。电平变化中断引脚。
PGC			I/O	ST	在线调试器和 ICSP 编程时钟引脚。
RB7/PGD	28	28			
RB7			I/O	TTL	数字 I/O。电平变化中断引脚。
PGD			I/O	ST	在线调试器和 ICSP 编程数据引脚。

注： TTL=TTL 兼容输入 CMOS=CMOS 兼容输入或输出
ST=CMOS 电平的施密特触发器输入 I= 输入
O= 输出 P= 电源
OD= 漏极开路 （没有二极管接到 V_{DD}）

表 **1-2**： **PIC18F2X2** 引脚排列 **I/O** 说明 （续）

引脚名	引脚号		引脚类型	缓冲器类型	说明
	DIP	SOIC			
					PORTC 是双向 I/O 口。
RC0/T1OSO/T1CKI	11	11			
RC0			I/O	ST	数字 I/O。
T1OSO			O	—	Timer1 晶振输出。
T1CKI			I	ST	Timer1/Timer3 外部时钟输入。
RC1/T1OSI/CCP2	12	12			
RC1			I/O	ST	数字 I/O。
T1OSI			I	CMOS	Timer1 晶振输入。
CCP2			I/O	ST	Capture2 输入、 Compare2 输出、 PWM2 输出。
RC2/CCP1	13	13			
RC2			I/O	ST	数字 I/O。
CCP1			I/O	ST	Capture1 输入 /Compare1 输出 /PWM1 输出。
RC3/SCK/SCL	14	14			
RC3			I/O	ST	数字 I/O。
SCK			I/O	ST	SPI 模式的同步串行时钟输入 / 输出。
SCL			I/O	ST	I^{2}C 模式的同步串行时钟输入 / 输出。
RC4/SDI/SDA	15	15			
RC4			I/O	ST	数字 I/O。
SDI			I	ST	SPI 模式的数据输入。
SDA			I/O	ST	I^{2}C 模式的数据 I/O。
RC5/SDO	16	16			
RC5			I/O	ST	数字 I/O。
SDO			O	—	SPI 模式的数据输出。
RC6/TX/CK	17	17			
RC6			I/O	ST	数字 I/O。
TX			O	—	USART 异步发送。
CK			I/O	ST	USART 同步时钟 （参见有关 RX/DT 的信息）。
RC7/RX/DT	18	18			
RC7			I/O	ST	数字 I/O。
RX			I	ST	USART 异步接收。
DT			I/O	ST	USART 同步数据 （参见有关 TX/CK 的信息）。
VSS	8, 19	8, 19	P	—	逻辑和 I/O 引脚的参考地。
VSS	20	20	P	—	逻辑和 I/O 引脚的正向电源。

注： TTL=TTL 兼容输入 CMOS=CMOS 兼容输入或输出
ST=CMOS 电平的施密特触发器输入 I= 输入
O= 输出 P= 电源
OD= 漏极开路 （没有二极管接到 VDD）

表 **1-3**： **PIC18F2X2** 引脚排列 **I/O** 说明

引脚名	引脚号			引脚类型	缓冲器类型	说明
	DIP	**PLCC**	**TQFP**			
MCLR/VPP	1	2	18			主清零 （输入）或高电平 ICSP 编程使能引脚。
MCLR				I	ST	主清零 （复位）输入。此引脚为低电平时，器件复位 （低电平有效）。
VPP				I	ST	高电平 ICSP 编程使能引脚。
NC	—			—	—	这些引脚应该悬空。
OSC1/CLKI	13	14	30			晶振或外部时钟输入。
OSC1				I	ST	晶振输入或外部时钟源输入。配置为 RC 模式时带 ST 缓冲器，否则带 CMOS 缓冲器。
CLKI				I	CMOS	外部时钟源输入。总是与 OSC1 引脚功能复用。（参见有关 OSC1/CLKI 和 OSC2/CLKO 引脚的信息。）
OSC2/CLKO/RA6	14	15	31			晶振或时钟输出。
OSC2				O	—	晶振输出。在晶振模式时，该引脚与晶振或谐振器相连。
CLKO				O	—	在 RC 模式时， OSC2 引脚输出 CLKO 振荡信号，该信号是 OSC1 引脚上的振荡信号的 4 分频，等于指令周期。
RA6				I/O	TTL	通用 I/O 引脚。
						PORTA 是双向 I/O 口。
RA0/AN0	2	3	19			
RA0				I/O	TTL	数字 I/O。
AN0				I	模拟	模拟输入 0。
RA1/AN1	3	4	20			
RA1				I/O	TTL	数字 I/O。
AN1				I	模拟	模拟输入 1。
RA2/AN2/VREF-	4	5	21			
RA2				I/O	TTL	数字 I/O。
AN2				I	模拟	模拟输入 2。
VREF-				I	模拟	A/D 参考电压 （低）输入。
RA3/AN3/VREF+	5	6	22			
RA3				I/O	TTL	数字 I/O。
AN3				I	模拟	模拟输入 3。
VREF+				I	模拟	A/D 参考电压 （高）输入。
RA4/T0CKI	6	7	23			
RA4				I/O	ST/OD	数字 I/O。当配置为输出时，为漏极开路。
T0CKI				I	ST	Timer0 外部时钟输入。
RA5/AN4/SS/LVDIN	7	8	24			
RA5				I/O	TTL	数字 I/O。
AN4				I	模拟	模拟输入 4。
SS				I	ST	SPI 从模式选择输入。
LVDIN				I	模拟	低压检测输入。
RA6						（参见 OSC2/CLKO/RA6 引脚。）

注： TTL=TTL 兼容输入　　CMOS=CMOS 兼容输入或输出
ST=CMOS 电平的施密特触发器输入　　I= 输入
O= 输出　　P= 电源
OD= 漏极开路 （没有二极管 P 极接到 VDD）

表 **1-3**： **PIC18F2X2** 引脚排列 **I/O** 说明 （续）

引脚名	引脚号			引脚类型	缓冲器类型	说明
	DIP	PLCC	TQFP			
						PORTB 是双向 I/O 口。 PORTB 所有输入端都有可编程的内部弱上拉。
RB0/INT0	33	36	8			
RB0				I/O	TTL	数字 I/O。
INT0				I	ST	外部中断 0。
RB1/INT1	34	37	9			
RB1				I/O	TTL	数字 I/O。
INT1				I	ST	外部中断 1。
RB2/INT2	35	38	10			
RB2				I/O	TTL	数字 I/O。
INT2				I	ST	外部中断 2。
RB3/CCP2	36	39	11			
RB3				I/O	TTL	数字 I/O。
CCP2				I/O	ST	Capture2 输入、 Compare2 输出、 PWM2 输出。
RB4	37	41	14	I/O	TTL	数字 I/O。电平变化中断引脚。
RB5/PGM	38	42	15			
RB5				I/O	TTL	数字 I/O。电平变化中断引脚。
PGM				I/O	ST	低压 ICSP 编程使能引脚。
RB6/PGC	39	43	16			
RB6				I/O	TTL	数字 I/O。电平变化中断引脚。
PGC				I/O	ST	在线调试器和 ICSP 编程时钟引脚。
RB7/PGD	40	44	17			
RB7				I/O	TTL	数字 I/O。电平变化中断引脚。
PGD				I/O	ST	在线调试器和 ICSP 编程数据引脚。

注： TTL=TTL 兼容输入　　CMOS=CMOS 兼容输入或输出
ST=CMOS 电平的施密特触发器输入　　I= 输入
O= 输出　　P= 电源
OD= 漏极开路 （没有二极管 P 极接到 V_{DD}）

表 **1-3**： **PIC18F2X2** 引脚排列 **I/O** 说明 （续）

引脚名	引脚号 DIP	PLCC	TQFP	引脚类型	缓冲器类型	说明
						PORTC 是双向 I/O 口。
RC0/T1OSO/T1CKI	15	16	32			
RC0				I/O	ST	数字 I/O。
T1OSO				O	—	Timer1 晶振输出。
T1CKI				I	ST	Timer1/Timer3 外部时钟输入。
RC1/T1OSI/CCP2	16	18	35			
RC1				I/O	ST	数字 I/O。
T1OSI				I	CMOS	Timer1 晶振输入。
CCP2				I/O	ST	Capture2 输入、 Compare2 输出、 PWM2 输出。
RC2/CCP1	17	19	36			
RC2				I/O	ST	数字 I/O。
CCP1				I/O	ST	Captuer1 输入 /Compare1 输出 /PWM1 输出。
RC3/SCK/SCL	18	20	37			
RC3				I/O	ST	数字 I/O。
SCK				I/O	ST	SPI 模式的同步串行时钟输入 / 输出。
SCL				I/O	ST	I^2C 模式的同步串行时钟输入 / 输出。
RC4/SDI/SDA	23	25	42			
RC4				I/O	ST	数字 I/O。
SDI				I	ST	SPI 模式的数据输入。
SDA				I/O	ST	I^2C 模式的数据 I/O。
RC5/SDO	24	26	43			
RC5				I/O	ST	数字 I/O。
SDO				O	—	SPI 模式的数据输出。
RC6/TX/CK	25	27	44			
RC6				I/O	ST	数字 I/O。
TX				O	—	USART 异步发送。
CK				I/O	ST	USART 同步时钟 （参见有关 RX/DT 的信息）。
RC7/RX/DT	26	29	1			
RC7				I/O	ST	数字 I/O。
RX				I	ST	USART 异步接收。
DT				I/O	ST	USART 同步数据 （参见有关 TX/CK 的信息）。

注： TTL=TTL 兼容输入
ST=CMOS 电平的施密特触发器输入
O= 输出
OD= 漏极开路 （没有二极管 P 极接到 VDD）
CMOS=CMOS 兼容输入或输出
I= 输入
P= 电源

表 **1-3**： **PIC18F2X2** 引脚排列 **I/O** 说明 （续）

引脚名	引脚号			引脚类型	缓冲器类型	说明
	DIP	**PLCC**	**TQFP**			
						PORTD 是双向 I/O 口，或是可以和微处理器端口接口的并行从动端口 （PSP）。当 PSP 模块使能时，这些引脚带有 TTL 输入缓冲器。
RD0/PSP0	19	21	38	I/O	ST TTL	数字 I/O。 并行从动端口数据。
RD1/PSP1	20	22	39	I/O	ST TTL	数字 I/O。 并行从动端口数据。
RD2/PSP2	21	23	40	I/O	ST TTL	数字 I/O。 并行从动端口数据。
RD3/PSP3	22	24	41	I/O	ST TTL	数字 I/O。 并行从动端口数据。
RD4/PSP4	27	30	2	I/O	ST TTL	数字 I/O。 并行从动端口数据。
RD5/PSP5	28	31	3	I/O	ST TTL	数字 I/O。 并行从动端口数据。
RD6/PSP6	29	32	4	I/O	ST TTL	数字 I/O。 并行从动端口数据。
RD7/PSP7	30	33	5	I/O	ST TTL	数字 I/O。 并行从动端口数据。
						PORTE 是双向 I/O 口。
RE0/$\overline{RD}$/AN5	8	9	25	I/O		
RE0					ST	数字 I/O。
$\overline{RD}$					TTL	并行从动端口的读控制 （参见有关 $\overline{WR}$ 和 $\overline{CS}$ 引脚的信息）。
AN5					模拟	模拟输入 5。
RE1/$\overline{WR}$/AN6	9	10	26	I/O		
RE1					ST	数字 I/O。
$\overline{WR}$					TTL	并行从动端口的写控制 （参见有关 $\overline{CS}$ 和 $\overline{RD}$ 引脚的信息）。
AN6					模拟	模拟输入 6。
RE2/$\overline{CS}$/AN7	10	11	27	I/O		
RE2					ST	数字 I/O。
$\overline{CS}$					TTL	并行从动端口的片选控制 （参见有关 $\overline{RD}$ 和 $\overline{WR}$ 引脚的信息）。
AN7					模拟	模拟输入 7。
V_{SS}	12, 31	13, 34	6, 29	P	—	逻辑和 I/O 引脚的参考地。
V_{DD}	11, 32	12, 35	7, 28	P	—	逻辑和 I/O 引脚的正向电源。

注： TTL=TTL 兼容输入 CMOS=CMOS 兼容输入或输出
ST=CMOS 电平的施密特触发器输入 I= 输入
O= 输出 P= 电源
OD= 漏极开路 （没有二极管 P 极接到 V_{DD}）

2.0 振荡器配置

2.1 振荡器类型

PIC18FXX2 可以在八种振荡模式下工作。用户可以通过对三个配置位（FOSC2、FOSC1 和 FOSC0）编程来选择其中一种模式：

1. LP 低功耗晶体
2. XT 晶振 / 谐振器
3. HS 高速晶体 / 谐振器
4. HS ＋ PLL 高速晶体/谐振器，允许使用锁相环
5. RC 外部电阻 / 电容振荡模式
6. RCIO 外部电阻 / 电容振荡模式，使用 I/O 引脚
7. EC 外部时钟振荡模式
8. ECIO 外部时钟振荡模式，使用 I/O 引脚

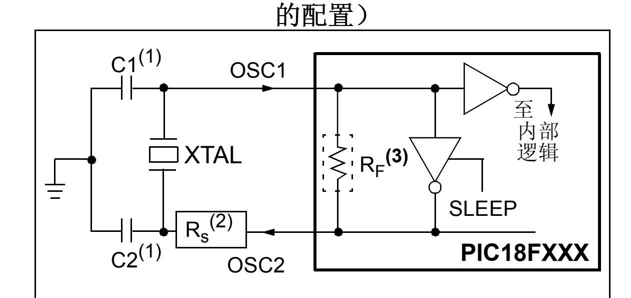
2.2 晶体振荡器 / 陶瓷谐振器

在 XT、LP、HS 或 HS+PLL 振荡模式下，OSC1 和 OSC2 引脚连接一个晶体谐振器或陶瓷谐振器以产生振荡。引脚连接方式如图 2-1 所示。

PIC18FXX2 振荡器的设计要求使用一个平行切割的晶体。

注：	使用顺序切割的晶体，可能使振荡器产生的频率超出晶体制造厂商所给的参数范围。

图 2-1： 晶振 / 陶瓷谐振器的运行（HS、XT 或 LP 振荡器模式的配置）



注 1：查阅表 2-1 和表 2-2 可以获得 C1 和 C2 的推荐值。
2：对于 AT 条形切割（AT Strip Cut）的晶体，可能需要一个串联电阻（R_S）。
3：R_f 的值根据振荡器模式而定。

表 2-1： 陶瓷谐振器电容的选择

测试范围：			
模式	频率	C1	C2
XT	455 kHz 2.0 MHz 4.0 MHz	68 - 100 pF 15 - 68 pF 15 - 68 pF	68 -00 pF 15-68 pF 15-68 pF
HS	8.0 MHz 16.0 MHz	10-68 pF 10-22 pF	10-68 pF 10-22 pF
这些值仅供设计参考。 请参见表下方的注释。			
所使用的谐振器：			
455 kHz	Panasonic EFO-A455K04B		± 0.3%
2.0 MHz	Murata Erie CSA2.00MG		± 0.5%
4.0 MHz	Murata Erie CSA2.00MG		± 0.5%
8.0 MHz	Murata Erie CSA2.00MG		± 0.5%
16.0 MHz	Murata Erie CSA2.00MG		± 0.5%
所用的谐振器都没有内置电容。			

注 1：电容值越大，振荡器越稳定，但起振时间越长。
2：当工作电压 Vdd 低于 3V，或当在任意电压下使用某些陶瓷谐振器时，可能需要采用高增益的 HS 模式，试着降低谐振器的频率，或采用晶体振荡器。
3：由于每个谐振器/晶体都有其自身的特性，用户应向谐振器 / 晶体厂商咨询外围元件的适当值，或验证振荡器的实际性能。

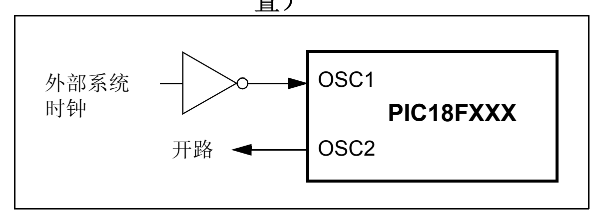
表 **2-2**： 晶体振荡器的电容选择

测试范围：			
模式	频率	**C1**	**C2**
LP	32.0 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	22-68 pF	22-68 pF
	1.0 MHz	15 pF	15 pF
	4.0 MHz	15 pF	15 pF
HS	4.0 MHz	15 pF	15 pF
	8.0 MHz	15 - 33 pF	15 - 33 pF
	20.0 MHz	15 - 33 pF	15 - 33 pF
	25.0 MHz	15 - 33 pF	15 - 33 pF
这些值仅供设计参考。 请参见表下方的注释。			
所使用的晶体			
32.0 kHz	Epson C-001R32.768K-A	± 20 PPM	
200 kHz	STD XTL 200.000KHz	± 20 PPM	
1.0 MHz	ECS ECS-10-13-1	± 50 PPM	
4.0 MHz	ECS ECS-40-20-1	± 50 PPM	
8.0 MHz	Epson CA-301 8.000M-C	± 30 PPM	
20.0 MHz	Epson CA-301 8.000M-C	± 30 PPM	

注 **1**：电容值越大，振荡器越稳定，但起振时间越长。
2：为避免对低驱动规格的晶体造成过驱动，在 HS 和 XT 模式下，可能需要串联电阻 Rs。
3：由于每个谐振器/晶体都有其自身的特性，用户应向谐振器 / 晶体厂商咨询外围元件的适当值，或验证振荡器的实际性能。

在 HS、XT 和 LP 模式下，也可以在 OSC1 引脚处连接一个外部时钟源，如图 2-2 所示。

图 **2-2**： 外部时钟输入运行 （**HS**、**XT** 或 **LP** 振荡器模式的配置）



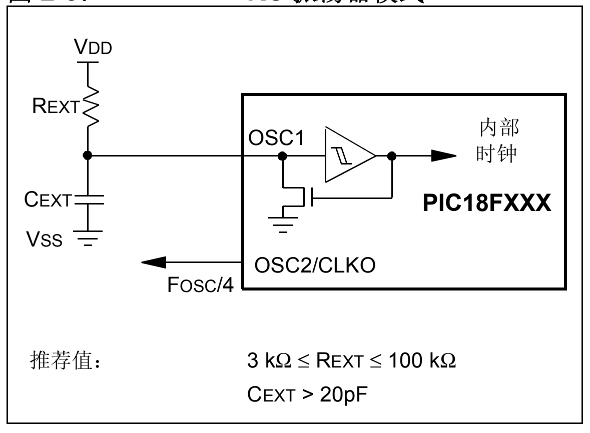
2.3 RC 振荡器

对于对时序要求不高的应用，选择 RC 和 RCIO 器件可以节约额外成本。 RC 振荡器频率是电源电压、电阻（REXT）、电容 （CEXT）和工作温度的函数。另外，由于正常的制造工艺参数的差异，每个器件的振荡频率也会有所不同。而不同封装的引线，其电容不同，也会影响振荡频率，特别是 CEXT 值较小时。用户还需要考虑由于外接电阻 R 和电容 C 的公差所带来的影响。图 2-3 显示了如何外接 R/C 电路。

在 RC 振荡器模式，振荡频率的 4 分频信号可由 OSC2 引脚输出。这个信号可用作测试或同步其他逻辑单元。

注： 如果实际应用中不需要使用振荡频率的4分频信号，则推荐使用 RCIO 模式以节省电流。

图 **2-3**： **RC** 振荡器模式



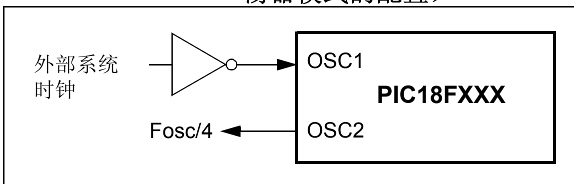
RCIO 振荡器模式类似于 RC 模式，不同之处在于 OSC2 引脚变成一个额外的通用 I/O 引脚。该 I/O 引脚变成 PORTA （RA6）的第 6 位。

2.4 外部时钟输入

EC 和 ECIO 振荡器模式要求 OSC1 引脚与一个外部时钟源相连。在这两种模式下，关掉了 OSC1 和 OSC2 之间的反馈器件以节省电流。在上电复位之后或从休眠模式下恢复时，不存在振荡器起振时间。

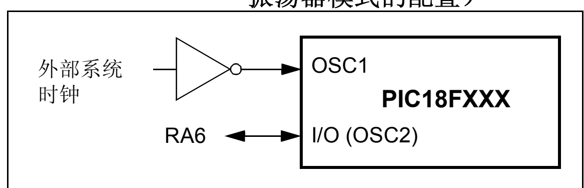
在 EC 振荡器模式，振荡频率的 4 分频信号可由 OSC2 引脚输出。这个信号可用作测试或同步其他逻辑单元。图 2-4 显示了 EC 振荡器模式的引脚连接。

图 2-4： 外部时钟输入运行 （EC 振荡器模式的配置）



ECIO 振荡器模式的工作方式类似于 EC 模式，不同之处在于 OSC2 引脚变成了一个额外的通用 I/O 引脚。该 I/O 引脚变成 PORTA（RA6）的第 6 位。图 2-5 显示了 ECIO 振荡器模式下引脚的连接。

图 2-5： 外部时钟输入运行 （ECIO 振荡器模式的配置）



2.5 HS/PLL

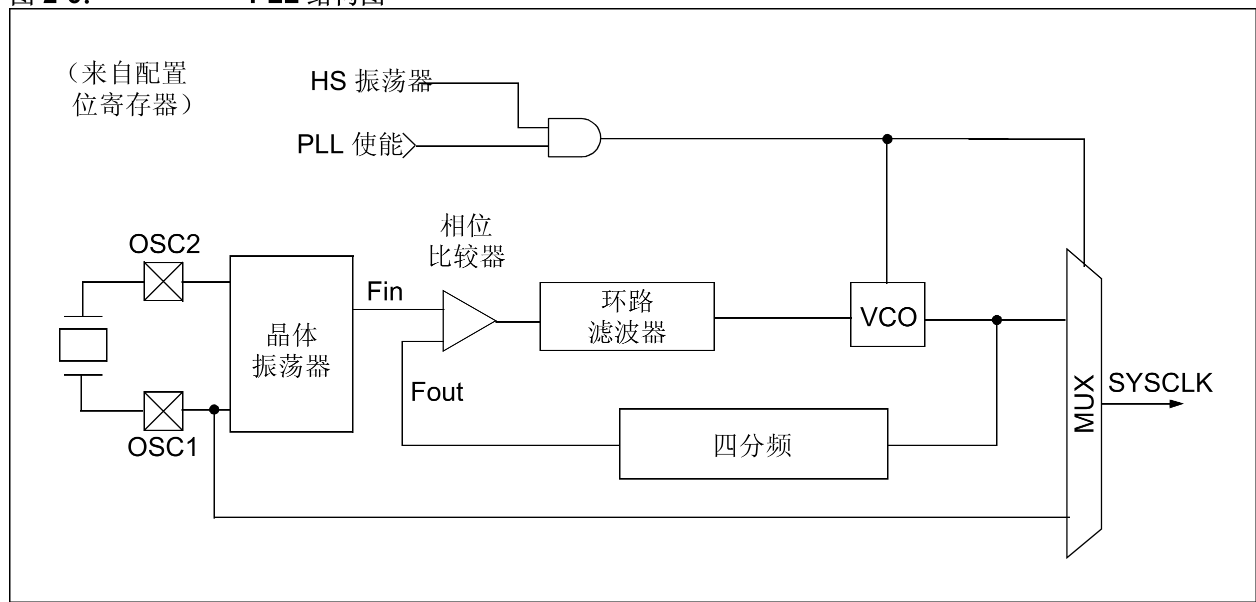
对于想把输入晶体振荡器信号变为四倍频的用户而言，锁相环电路提供了一种可编程的方法。对于 10 MHz 的输入时钟频率，内部时钟频率倍频为 40 MHz。这对那些介意因高频晶体而导致 EMI 的用户有用。

PLL 只有在通过程序将振荡模式位指定为 HS 模式时才可使用。如果程序中指定的是其他振荡器模式，那么 PLL 不可用，系统时钟将直接自 OSC1 引脚输入。

PLL 是 FOSC<2:0> 配置位所决定的一种模式。振荡器模式在器件编程期间指定。

PLL 锁定定时器用于确保 PLL 在器件开始运行前已经锁定。 PLL 锁定定时器有一个超时，称为 TPLL。

图 2-6： PLL 结构图

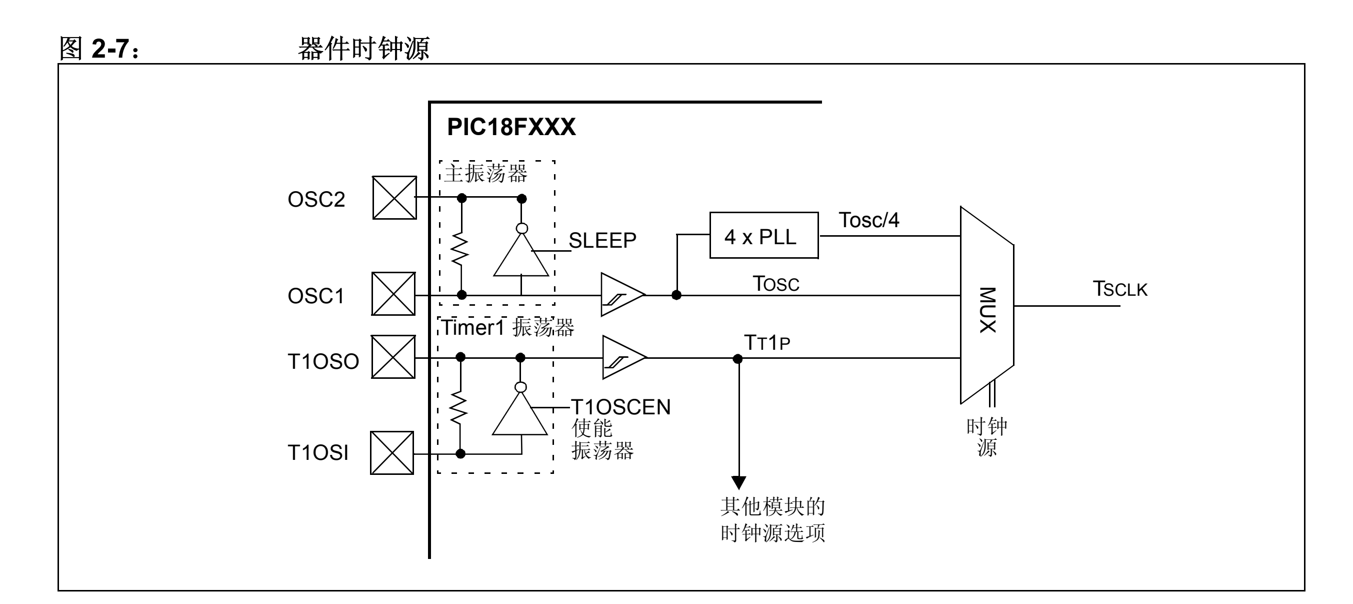


2.6 振荡器切换功能

PIC18FXX2 器件支持系统时钟源从主振荡器切换到备用低频时钟源。对于 PIC18FXX2 器件而言，这个备用时钟源就是 Timer1 振荡器。如果 Timer1 振荡器的引脚连接了一个低频晶体（例如， 32 kHz），并且 Timer1 振荡器被使能，则该器件可以切换到低功耗运行模式。

图 2-7 表示了系统时钟源的结构图。在程序中将配置寄存器 1H 中的振荡器切换使能位（OSCSEN）置 0 即可启用时钟切换功能。在已擦除器件中，时钟切换是禁止的。可以查看第 11.0 节以获取关于 Timer1 振荡器的详细信息。可以查看第 19.0 节以获取配置寄存器的详细信息。

图 2-7： 器件时钟源



2.6.1 系统时钟切换位

系统时钟源之间的切换是通过软件控制来实现的。系统时钟切换位 SCS （OSCCON<0>）控制时钟切换。当 SCS 位为 0 时，系统的时钟源是主振荡器，主振荡器由配置寄存器 1H 的 FOSC 配置位选择决定。当 SCS 位置 1 时，系统时钟源是 Timer1 振荡器。当出现任何形式的复位时， SCS 位被清零。

注：	必须使能 Timer1 振荡器并运行起来才能实现系统时钟源的切换。通过将 Timer1 控制寄存器（T1CON）的 T1OSCEN 位置 1 来使能 Timer1 振荡器。如果 Timer1 振荡器未被使能，则忽略对 SCS 位的任何写操作（SCS 位被强制清零），主振荡器仍会继续用作系统时钟源。

寄存器 2-1： OSCCON 寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-1
—	—	—	—	—	—	—	SCS
bit 7							bit 0

bit 7-1 **未实现**：读为 0

bit 0 **SCS**：系统时钟切换位

如果 $\overline{\text{OSCSEN}}$ 配置位为 0 以及 T1OSCEN 位置 1：
1 = 切换到 Timer1 振荡器 / 时钟引脚
0 = 采用主振荡器 / 时钟输入引脚

如果 $\overline{\text{OSCSEN}}$ 和 T1OSCEN 在其他状态：
该位被强制清零

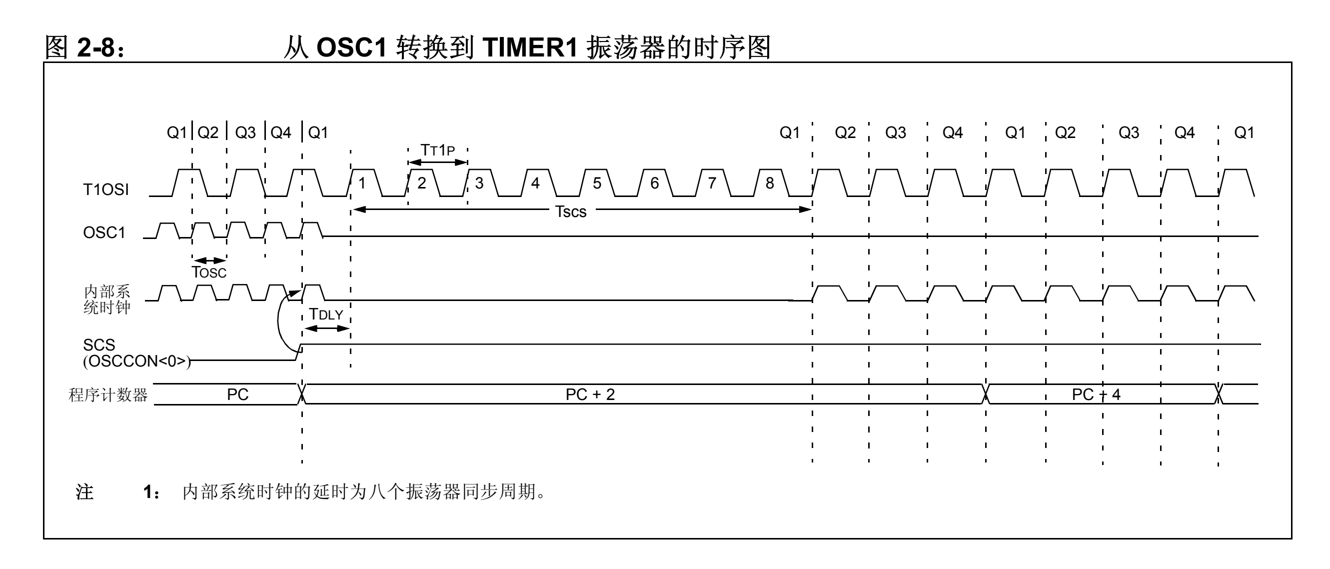
图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
- n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知位

2.6.2 振荡器转换

PIC18FXX2 器件中具有防止在振荡源之间进行切换时发生失灵的电路。这个电路实际上会等待新时钟源的八个上升沿，然后处理器才切换过去。这就保证了新时钟源是稳定的，以及脉冲宽度不小于这两个时钟源最窄的脉冲宽度。

从主振荡器转换到 Timer1 振荡器的时序图如图 2-8 所示。假设 Timer1 振荡器一直处于运行中。在 SCS 位置 1 以后，在 Q1 周期下次出现时处理器会被冻结。经过 Timer1 振荡器的八个同步周期之后，恢复运行。在同步周期之后不需要额外的延时。

图 2-8：　从 OSC1 转换到 TIMER1 振荡器的时序图

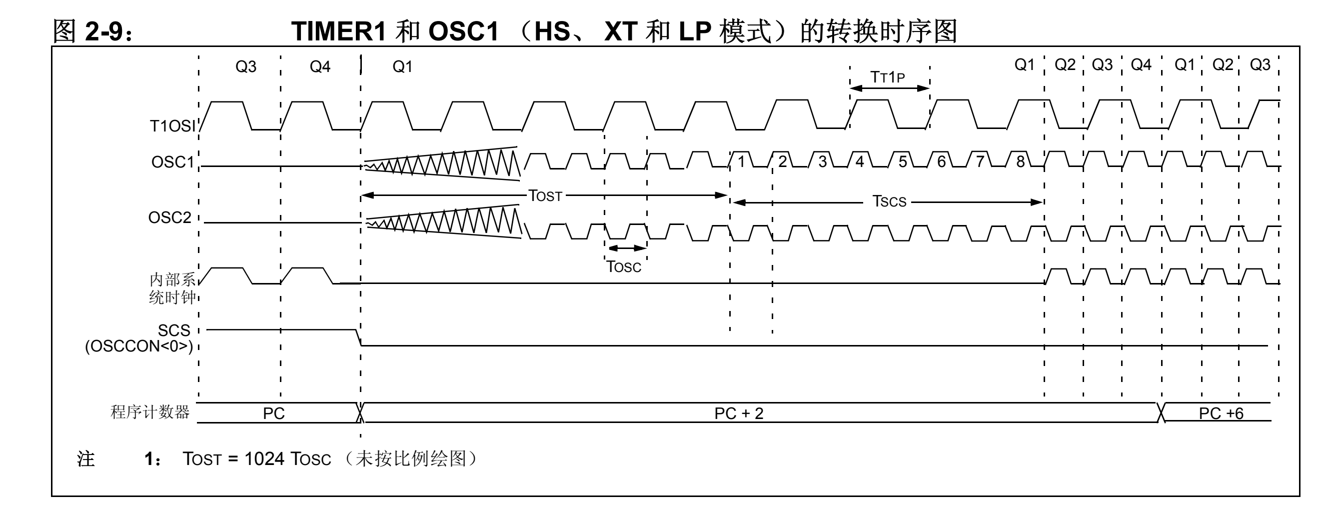


注　1：内部系统时钟的延时为八个振荡器同步周期。

从 Timer1 振荡器切换到主振荡器的事件顺序取决于主振荡器的模式。除主振荡器的八个时钟周期外，还需要额外的延时。

如果主振荡器配置为外部晶振模式（HS、XT 和 LP），那么将在一个振荡器起振时间（TOST）之后开始转换。从 Timer1 振荡器转换到 HS、XT 和 LP 模式下主振荡器的时序图如图 2-9 所示。

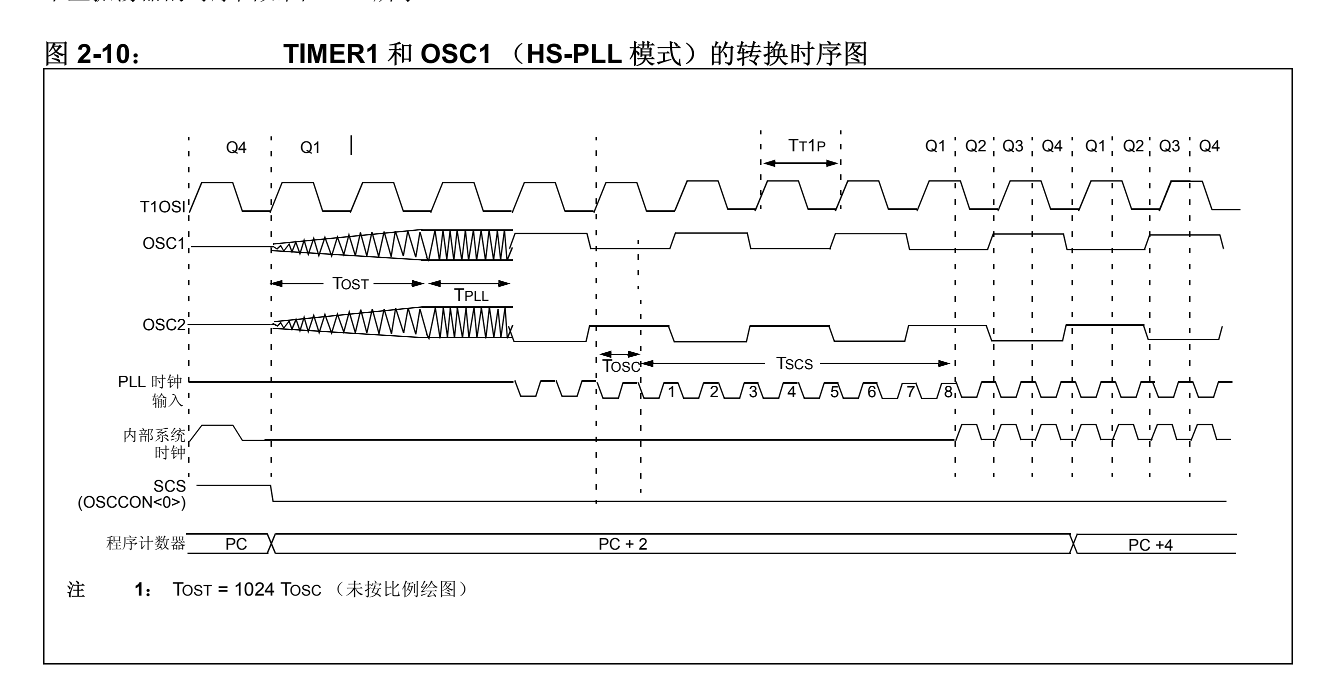
图 2-9：　TIMER1 和 OSC1（HS、XT 和 LP 模式）的转换时序图



注　1：TOST = 1024 TOSC（未按比例绘图）

如果主振荡器配置为 HS-PLL 模式，则需要振荡器起振时间 （TOST）再加上额外的 PLL 超时 （TPLL）之后开始转换。 PLL 超时通常是 2 ms，并允许 PLL 锁定在主振荡器频率上。从 Timer1 振荡器转换到 HS-PLL 模式下主振荡器的时序图如图 2-10 所示。

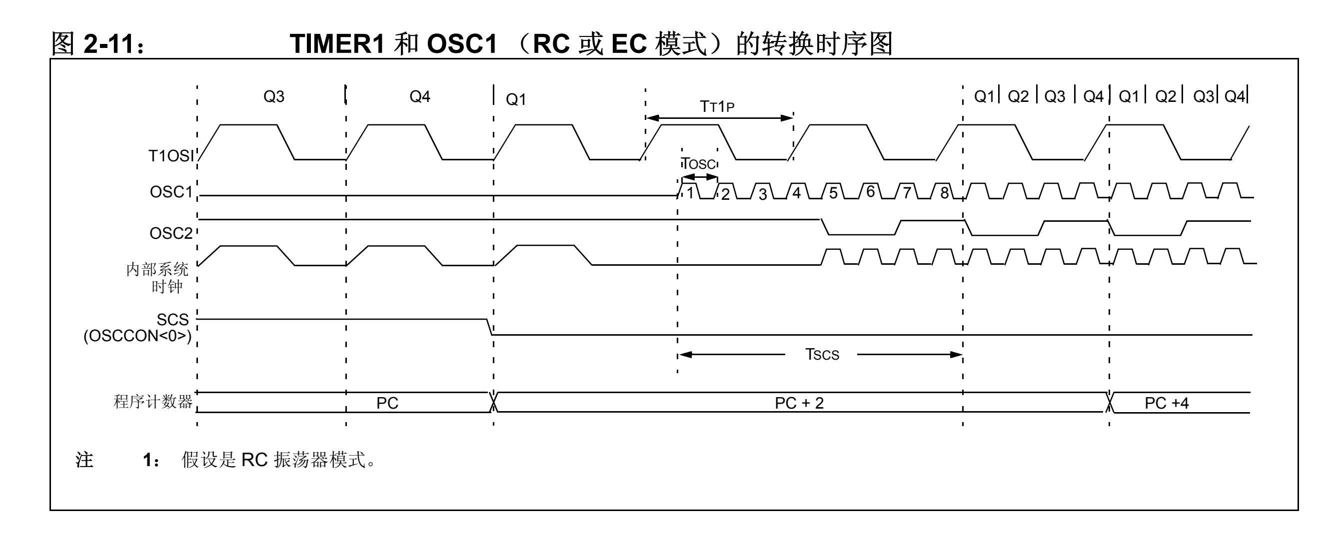
图 2-10： TIMER1 和 OSC1 （HS-PLL 模式）的转换时序图



注 1： TOST = 1024 TOSC （未按比例绘图）

如果主振荡器配置为 RC、 RCIO、 EC 或 ECIO 模式，就不会发生振荡器起振超时。在主振荡器计数达到八个周期之后，恢复运行。从 Timer1 振荡器转换到 RC、RCIO、EC 和 ECIO 模式下主振荡器的时序图如图 2-10 所示。

图 2-11： TIMER1 和 OSC1 （RC 或 EC 模式）的转换时序图



注 1： 假设是 RC 振荡器模式。

2.7 休眠模式对片内振荡器的影响

当器件执行一条SLEEP指令后，片内时钟和振荡器均被关闭，器件状态保持为指令周期的起始状态（Q1 状态）。随着振荡器的关闭，OSC1 和 OSC2 引脚的信号将会停止振荡。由于没有了晶体管的开关电流，使休眠模式器件的电流消耗达到最低值（仅有泄漏电流）。使能任何在休眠状态仍工作的片内功能，都将增加休眠状态的电流消耗。用户可通过外部复位、看门狗定时器复位或中断将器件从休眠状态唤醒。

表 2-3： 休眠模式下 OSC1 和 OSC2 引脚状态

OSC 模式	OSC1 引脚	OSC2 引脚
RC	悬空，经外部电阻上拉为高电平	逻辑低电平
RCIO	悬空，经外部电阻上拉为高电平	配置为 PORTA 的第 6 位
ECIO	悬空	配置为 PORTA 的第 6 位
EC	悬空	逻辑低电平
LP、 XT 和 HS	反馈反相器被禁用，处于静态电平	反馈反相器被禁用，处于静态电平

注： 关于由休眠和 $\overline{\text{MCLR}}$ 复位引起的超时，请参阅 **“复位”** 一节的表 3-1。

2.8 上电延时

有两个定时器控制上电延时，这样大部分的应用无需外接复位电路。这些延时确保在器件电源和时钟达到稳定之前，器件始终处于复位状态。更多关于复位操作的信息，请参阅第 3.0 节。

第一个定时器是是上电延时定时器（PWRT），它只为上电（POR 和 BOR）可选地提供固定的 72ms（标准值）延时。第二个定时器是振荡器起振定时器（OST），可以在晶体振荡器达到稳定状态前，使单片机始终处于复位状态。

如果使能 PLL（HS/PLL 振荡器模式），则上电复位后的超时序列与其他振荡器模式的不同。超时序列如下：首先， PWRT 超时是在 POR 延时超时后被调用的。然后，调用振荡器起振定时器（OST）。但是，这点时间仍然不足以使 PLL 锁定在高频上。PWRT 定时器用于提供一个额外的固定的 2 ms（标准值）超时，使得 PLL 有足够的时间锁定在输入时钟频率上。

3.0 复位

PIC18FXXX 有以下几种复位方式：

a) 上电复位 （POR）
b) 正常工作状态下的 MCLR 复位
c) 休眠状态下的 MCLR 复位
d) 看门狗定时器（WDT）复位（正常工作状态下）
e) 可编程的欠压复位 （BOR）
f) RESET 指令
g) 堆栈满复位
h) 堆栈下溢复位

大部分寄存器不受复位的影响。在上电复位 （POR）时，它们的状态未知，且不会被其他任何复位所改变。而另一些寄存器通过上电复位、MCLR、WDT 复位、欠压复位和休眠状态下的 MCLR 复位及 RESET 指令被强制进入复位状态。

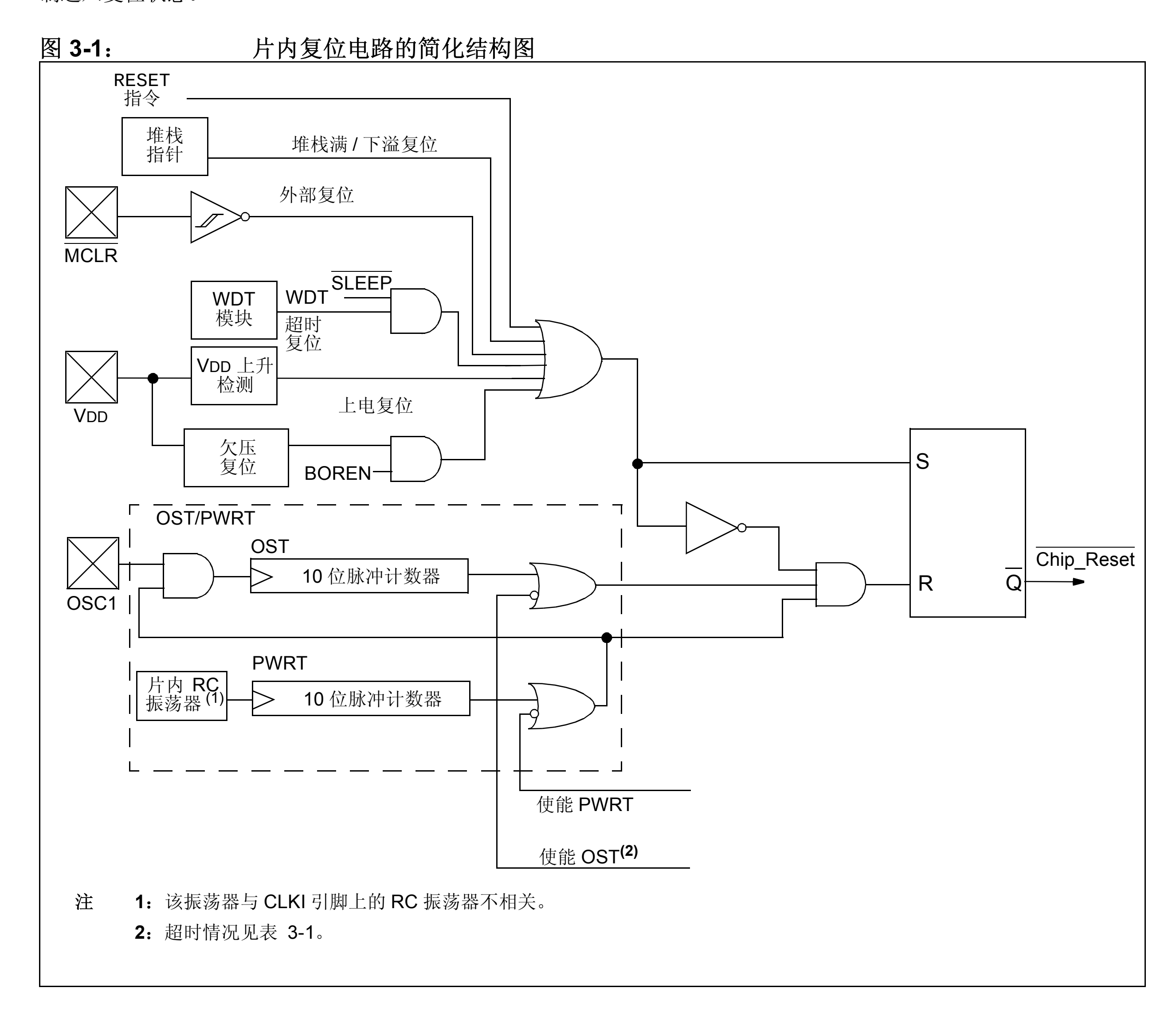
因为 WDT 唤醒被看作是恢复正常运行的操作，所以大部分寄存器不受 WDT 唤醒的影响。RCON 寄存器中的状态位：RI、TO、PD、POR 和 BOR 在不同的复位情况下分别被置 1 或清零，如表 3-2 所示。这些状态位在软件中用于判断复位的类型。表 3-3 对所有寄存器的复位状态作了完整的介绍。

图 3-1 显示了一个片内复位电路的简化结构图。

增强型 MCU 器件在 MCLR 复位信号的传输路径上有一个 MCLR 噪声滤波器。该滤波器可以检测并滤掉小脉冲信号。

WDT和任何内部复位不会将MCLR引脚驱动为低电平。

图 3-1： 片内复位电路的简化结构图

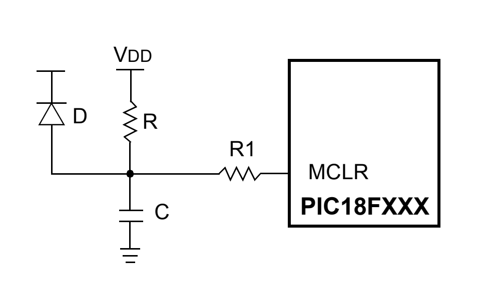


3.1 上电复位

检测到 VDD 上升信号时，会在片上产生一个上电复位（POR）脉冲。要利用 POR 电路，只需直接（或通过一个电阻）将 MCLR 引脚与 VDD 相连。这样可以避免使用通常产生上电复位延时所需的外部 RC 元件。参数 D004 中规定了 VDD 的最小上升率。图 3-2 是缓慢上升时间的电路。

当器件开始正常工作（即退出复位状态）时，其工作参数（电压、频率和温度等）必须在相应的工作范围内，否则将不能正常工作。如果不满足这些条件，器件必须保持在复位状态，直到满足工作条件为止。

图 3-2： 外部上电复位电路（VDD 缓慢上电）



注 1：只有当 VDD 上电速率过慢时，才需要外部上电复位电路。当 VDD 掉电时，二极管 D 使电容迅速放电。

2：建议 R < 40 kΩ，确保电阻 R 两端压降符合器件的电气规范参数。

3：R1 为 100Ω 到 1 kΩ 将限制电流从外部电容 C 流入 MCLR，以防万一发生因静电放电（Electrostatic Discharge，ESD）或过度电应力（Electrical Overstress，EOS）而导致 MCLR/VPP 引脚损坏。

3.2 上电延时定时器（PWRT）

上电延时定时器（PWRT）只对 POR 上电提供固定的标准延时（见参数 33）。上电延时定时器工作在内部 RC 振荡器上。只要 PWRT 有效，器件就保持在复位状态。PWRT 延时使 VDD 上升到一个适当的电压。用一个配置位可以使能 / 禁止 PWRT。

根据不同的 VDD、环境温度和制造工艺，不同芯片的上电延时也各不相同。详情请参阅 DC 参数 D033。

3.3 振荡器起振定时器（OST）

在 PWRT 延时结束以后（见参数 32），振荡器起振定时器（OST）提供了一个从 OSC1 输入的 1024 振荡周期的延时。这可以确保晶体振荡器或谐振器已经起振并趋于稳定。

只有在 XT、LP 和 HS 振荡器模式下，并且仅在上电复位或从休眠状态中被唤醒时，OST 延时定时器才开始工作。

3.4 PLL 锁定延时

使能 PLL 之后，上电复位后的延时序列与其他振荡器模式不同。上电延时定时器的一部分将提供足够的固定延时，让 PLL 有足够的时间锁定在主振荡器的频率上。PLL 锁定延时（TPLL）通常为 2ms，且在振荡器起振延时（OST）后发生。

3.5 欠压复位（BOR）

BOREN 配置位可以禁止（通过清零 / 编程）或使能（通过置 1）欠压复位（BOR）电路。如果 VDD 电压低于参数 D005的持续时间超过参数 35，这种欠压状况将使芯片复位。如果 VDD 电压低于参数 D005 的持续时间不超过参数 35，就不一定会发生复位。芯片保持欠压复位状态，直至 VDD 电压上升到 BVDD 以上。如果使能上电延时定时器，则它将在 VDD 电压上升至 BVDD 以上之后开始工作；并使芯片在下一个延时（见参数 33）期间保持复位状态。如果上电延时定时器运行时 VDD 电压降到 BVDD 以下，芯片将重新回到欠压复位状态，且将初始化上电延时定时器。一旦 VDD 电压上升到 BVDD 以上，上电延时定时器将再执行一个延时。

3.6 延时时序

上电延时时序为：首先，在 POR 延时到了之后进入 PWRT 延时。然后，OST 被激活。总延时时间将取决于振荡器的配置和 PWRT 的状态。例如，当 RC 模式下禁止PWRT时，绝对不会出现延时。图 3-3、图 3-4、图 3-5、图 3-6 和图 3-7 描绘了上电延时时序。

由 POR 脉冲引起的延时，如果 MCLR 保持足够长时间的低电平，则该延时将结束。此时立即执行将 MCLR 变为高电平（图 3-5）。这对于测试或同步并行的多个 PIC18FXXX 是非常有用的。

表 3-2 显示某些特殊功能寄存器的复位状态，而表 3-3 显示所有寄存器的复位状态。

表 3-1： 不同情况下的延时

振荡器配置	上电 (2)		欠压	从休眠中唤醒或振荡器切换
	$\overline{\text{PWRTE}}$ = 0	$\overline{\text{PWRTE}}$ = 1		
启用 PLL 的 HS(1)	72 ms + 1024 Tosc + 2ms	1024 Tosc + 2 ms	72 ms(2) + 1024 Tosc + 2 ms	1024 Tosc + 2 ms
HS， XT， LP	72 ms + 1024 Tosc	1024 Tosc	72 ms(2) + 1024 Tosc	1024 Tosc
EC	72 ms	—	72 ms(2)	—
外部 RC	72 ms	—	72 ms(2)	—

注 **1：** 4x PLL 锁定所需的标准时间为 2 ms。
2： 上电延时定时器标准延迟为 72 ms （如果可实现）。

寄存器 3-1： **RCON 寄存器位和位置**

R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
bit 7							bit 0

注 **1：** 关于位的定义请参阅第 4.14 节 （第 53 页）。

表 3-2： **RCON 寄存器的状态位、含义以及初始化状态**

状态	程序计数器	RCON 寄存器	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	STKFUL	STKUNF
上电复位	0000h	0--1 1100	1	1	1	0	0	u	u
正常工作状态下的 $\overline{\text{MCLR}}$ 复位	0000h	0--u uuuu	u	u	u	u	u	u	u
正常工作状态下的软件复位	0000h	0--0 uuuu	0	u	u	u	u	u	u
正常工作状态下的堆栈满复位	0000h	0--u uu11	u	u	u	u	u	u	1
正常工作状态下的堆栈下溢复位	0000h	0--u uu11	u	u	u	u	u	1	u
休眠状态下的 $\overline{\text{MCLR}}$ 复位	0000h	0--u 10uu	u	1	0	u	u	u	u
WDT 复位	0000h	0--u 01uu	1	0	1	u	u	u	u
WDT 唤醒	PC + 2	u--u 00uu	u	0	0	u	u	u	u
欠压复位	0000h	0--1 11u0	1	1	1	1	0	u	u
休眠时的中断唤醒	PC + 2(1)	u--u 00uu	u	1	0	u	u	u	u

图注： u= 不变， x= 未知， -= 未实现位，读为 0
注 **1：** 当芯片被中断唤醒且 GIEH 或 GIEL 位被置 1 时， PC 装入中断矢量 （0x000008h 或 0x000018h）。

表 3-3：　所有寄存器的初始化状态

寄存器	适用器件				上电复位，欠压复位	MCLR 复位 WDT 复位 RESET 指令 堆栈复位	通过 WDT 或中断唤醒
TOSU	242	442	252	452	---0 0000	---0 0000	---0 uuuu(3)
TOSH	242	442	252	452	0000 0000	0000 0000	uuuu uuuu(3)
TOSL	242	442	252	452	0000 0000	0000 0000	uuuu uuuu(3)
STKPTR	242	442	252	452	00-0 0000	uu-0 0000	uu-u uuuu(3)
PCLATU	242	442	252	452	---0 0000	---0 0000	---u uuuu
PCLATH	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
PCL	242	442	252	452	0000 0000	0000 0000	PC + 2(2)
TBLPTRU	242	442	252	452	--00 0000	--00 0000	--uu uuuu
TBLPTRH	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
TBLPTRL	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
TABLAT	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
PRODH	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
PRODL	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
INTCON	242	442	252	452	0000 000x	0000 000u	uuuu uuuu(1)
INTCON2	242	442	252	452	1111 -1-1	1111 -1-1	uuuu -u-u(1)
INTCON3	242	442	252	452	11-0 0-00	11-0 0-00	uu-u u-uu(1)
INDF0	242	442	252	452	N/A	N/A	N/A
POSTINC0	242	442	252	452	N/A	N/A	N/A
POSTDEC0	242	442	252	452	N/A	N/A	N/A
PREINC0	242	442	252	452	N/A	N/A	N/A
PLUSW0	242	442	252	452	N/A	N/A	N/A
FSR0H	242	442	252	452	---- xxxx	---- uuuu	---- uuuu
FSR0L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
WREG	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
INDF1	242	442	252	452	N/A	N/A	N/A
POSTINC1	242	442	252	452	N/A	N/A	N/A
POSTDEC1	242	442	252	452	N/A	N/A	N/A
PREINC1	242	442	252	452	N/A	N/A	N/A
PLUSW1	242	442	252	452	N/A	N/A	N/A

图注：　u= 不变， x= 未知， -= 未实现位，读为 0， q= 根据状态而变化。
阴影单元格表示不适用于指定器件。

注　1：INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。
2：当芯片被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断矢量 （0008h 或 0018h）。
3：当芯片被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、 TOSH 和 TOSL。 STKPTR 被修改为指向硬件堆栈中的下一个单元。
4：具体情况的复位值请参阅表 3-2。
5：PORTA、 LATA 和 TRISA 的 bit 6 只在 ECIO 和 RCIO 振荡器模式下被使能。在所有其他振荡器模式下，它们被禁止并读为 0。
6：PORTA、 LATA 和 TRISA 的 bit 6 并非在所有器件上都可用。不可用的时候，它们被读为 0。

表 3-3： 所有寄存器的初始化状态 （续）

寄存器	适用器件				上电复位，欠压复位	$\overline{MCLR}$ 复位 WDT 复位 RESET 指令 堆栈复位	通过 WDT 或中断唤醒
FSR1H	242	442	252	452	---- xxxx	---- uuuu	---- uuuu
FSR1L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	242	442	252	452	---- 0000	---- 0000	---- uuuu
INDF2	242	442	252	452	N/A	N/A	N/A
POSTINC2	242	442	252	452	N/A	N/A	N/A
POSTDEC2	242	442	252	452	N/A	N/A	N/A
PREINC2	242	442	252	452	N/A	N/A	N/A
PLUSW2	242	442	252	452	N/A	N/A	N/A
FSR2H	242	442	252	452	---- xxxx	---- uuuu	---- uuuu
FSR2L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	242	442	252	452	---x xxxx	---u uuuu	---u uuuu
TMR0H	242	442	252	452	0000 0000	uuuu uuuu	uuuu uuuu
TMR0L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	242	442	252	452	1111 1111	1111 1111	uuuu uuuu
OSCCON	242	442	252	452	---- ---0	---- ---0	---- ---u
LVDCON	242	442	252	452	--00 0101	--00 0101	--uu uuuu
WDTCON	242	442	252	452	---- ---0	---- ---0	---- ---u
RCON[4]	242	442	252	452	0--q 11qq	0--q qquu	u--u qquu
TMR1H	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	242	442	252	452	0-00 0000	u-uu uuuu	u-uu uuuu
TMR2	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
PR2	242	442	252	452	1111 1111	1111 1111	1111 1111
T2CON	242	442	252	452	-000 0000	-000 0000	-uuu uuuu
SSPBUF	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
SSPCON1	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
SSPCON2	242	442	252	452	0000 0000	0000 0000	uuuu uuuu

图注： u= 不变， x= 未知， -= 未实现位，读为 0， q= 根据状态而变化。
阴影单元格表示不适用于指定器件。

注 1：INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。
2：当芯片被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断矢量 （0008h 或 0018h）。
3：当芯片被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、 TOSH 和 TOSL。 STKPTR 被修改为指向硬件堆栈中的下一个单元。
4：具体情况的复位值请参阅表 3-2。
5：PORTA、 LATA 和 TRISA 的 bit 6 只在 ECIO 和 RCIO 振荡器模式下被使能。在所有其他振荡器模式下，它们被禁止并读为 0。
6：PORTA、 LATA 和 TRISA 的 bit 6 并非在所有器件上都可用。不可用的时候，它们被读为 0。

表 3-3： 所有寄存器的初始化状态 （续）

寄存器	适用器件				上电复位，欠压复位	MCLR 复位 WDT 复位 RESET 指令 堆栈复位	通过 WDT 或中断唤醒
ADRESH	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	242	442	252	452	0000 00-0	0000 00-0	uuuu uu-u
ADCON1	242	442	252	452	00-- 0000	00-- 0000	uu-- uuuu
CCPR1H	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	242	442	252	452	--00 0000	--00 0000	--uu uuuu
CCPR2H	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	242	442	252	452	--00 0000	--00 0000	--uu uuuu
TMR3H	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	242	442	252	452	0000 0000	uuuu uuuu	uuuu uuuu
SPBRG	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
RCREG	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
TXREG	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
TXSTA	242	442	252	452	0000 -010	0000 -010	uuuu -uuu
RCSTA	242	442	252	452	0000 000x	0000 000x	uuuu uuuu
EEADR	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
EEDATA	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
EECON1	242	442	252	452	xx-0 x000	uu-0 u000	uu-0 u000
EECON2	242	442	252	452	---- ----	---- ----	---- ----

图注： u= 不变， x= 未知， -= 未实现位，读为 0， q= 根据状态而变化。
阴影单元格表示不适用于指定器件。

注 **1：** INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。
2： 当芯片被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断矢量 （0008h 或 0018h）。
3： 当芯片被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、 TOSH 和 TOSL。 STKPTR 被修改为指向硬件堆栈中的下一个单元。
4： 具体情况的复位值请参阅表 3-2。
5： PORTA、 LATA 和 TRISA 的 bit 6 只在 ECIO 和 RCIO 振荡器模式下被使能。在所有其他振荡器模式下，它们被禁止并读为 0。
6： PORTA、 LATA 和 TRISA 的 bit 6 并非在所有器件上都可用。不可用的时候，它们被读为 0。

表 **3-3**： 所有寄存器的初始化状态 （续）

寄存器	适用器件				上电复位，欠压复位	MCLR 复位 WDT 复位 RESET 指令 堆栈复位	通过 WDT 或中断唤醒
IPR2	242	442	252	452	---1 1111	---1 1111	---u uuuu
PIR2	242	442	252	452	---0 0000	---0 0000	---u uuuu(1)
PIE2	242	442	252	452	---0 0000	---0 0000	---u uuuu
IPR1	242	442	252	452	1111 1111	1111 1111	uuuu uuuu
	242	442	252	452	-111 1111	-111 1111	-uuu uuuu
PIR1	242	442	252	452	0000 0000	0000 0000	uuuu uuuu(1)
	242	442	252	452	-000 0000	-000 0000	-uuu uuuu(1)
PIE1	242	442	252	452	0000 0000	0000 0000	uuuu uuuu
	242	442	252	452	-000 0000	-000 0000	-uuu uuuu
TRISE	242	442	252	452	0000 -111	0000 -111	uuuu -uuu
TRISD	242	442	252	452	1111 1111	1111 1111	uuuu uuuu
TRISC	242	442	252	452	1111 1111	1111 1111	uuuu uuuu
TRISB	242	442	252	452	1111 1111	1111 1111	uuuu uuuu
TRISA(5,6)	242	442	252	452	-111 1111(5)	-111 1111(5)	-uuu uuuu(5)
LATE	242	442	252	452	---- -xxx	---- -uuu	---- -uuu
LATD	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATC	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA(5,6)	242	442	252	452	-xxx xxxx(5)	-uuu uuuu(5)	-uuu uuuu(5)
PORTE	242	442	252	452	---- -000	---- -000	---- -uuu
PORTD	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	242	442	252	452	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA(5,6)	242	442	252	452	-x0x 0000(5)	-u0u 0000(5)	-uuu uuuu(5)

图注： u= 不变， x= 未知， -= 未实现位，读为 0， q= 根据状态而变化。
阴影单元格表示不适用于指定器件。

注 **1**：INTCONx 或 PIRx 寄存器中的一位或多位会受到影响 （引起唤醒）。
2：当芯片被中断唤醒且 GIEL 或 GIEH 位被置 1 时， PC 装入中断矢量 （0008h 或 0018h）。
3：当芯片被中断唤醒且 GIEL 或 GIEH 位被置 1 时，用 PC 的当前值更新 TOSU、 TOSH 和 TOSL。 STKPTR 被修改为指向硬件堆栈中的下一个单元。
4：具体情况的复位值请参阅表 3-2。
5：PORTA、 LATA 和 TRISA 的 bit 6 只在 ECIO 和 RCIO 振荡器模式下被使能。在所有其他振荡器模式下，它们被禁止并读为 0。
6：PORTA、 LATA 和 TRISA 的 bit 6 并非在所有器件上都可用。不可用的时候，它们被读为 0。

图 3-3： 上电延时时序（$\overline{\text{MCLR}}$ 引脚与 VDD 相连）

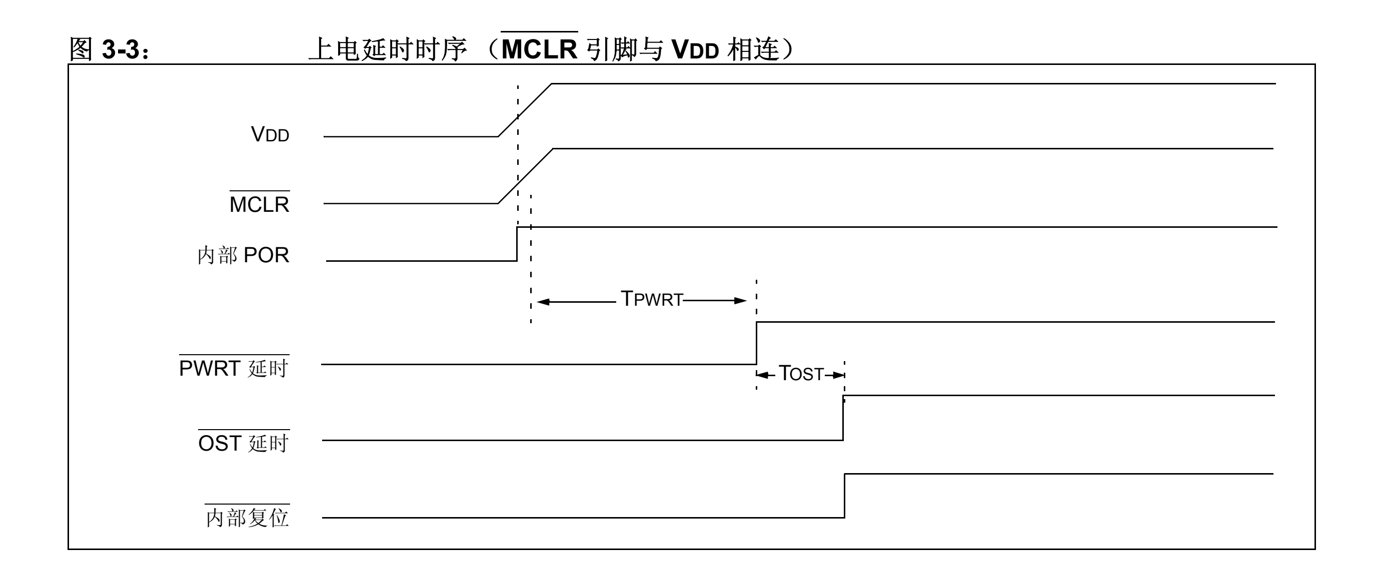


图 3-4： 上电延时时序（$\overline{\text{MCLR}}$ 引脚未与 VDD 相连）：情况 1

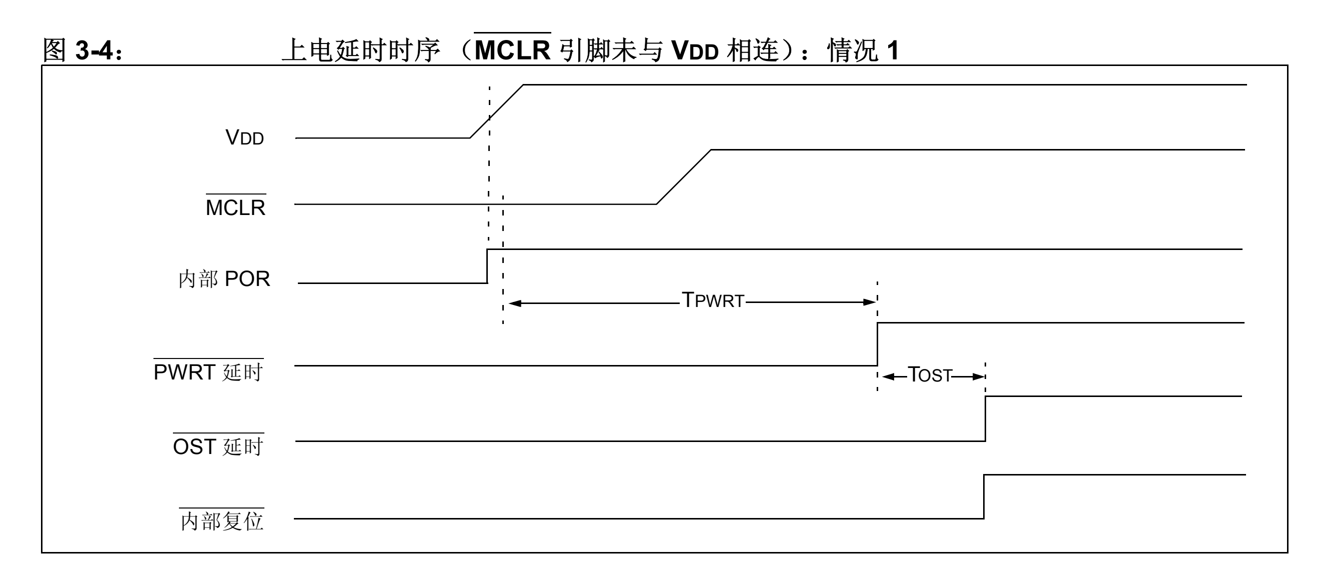


图 3-5： 上电延时时序（$\overline{\text{MCLR}}$ 引脚未与 VDD 相连）：情况 2

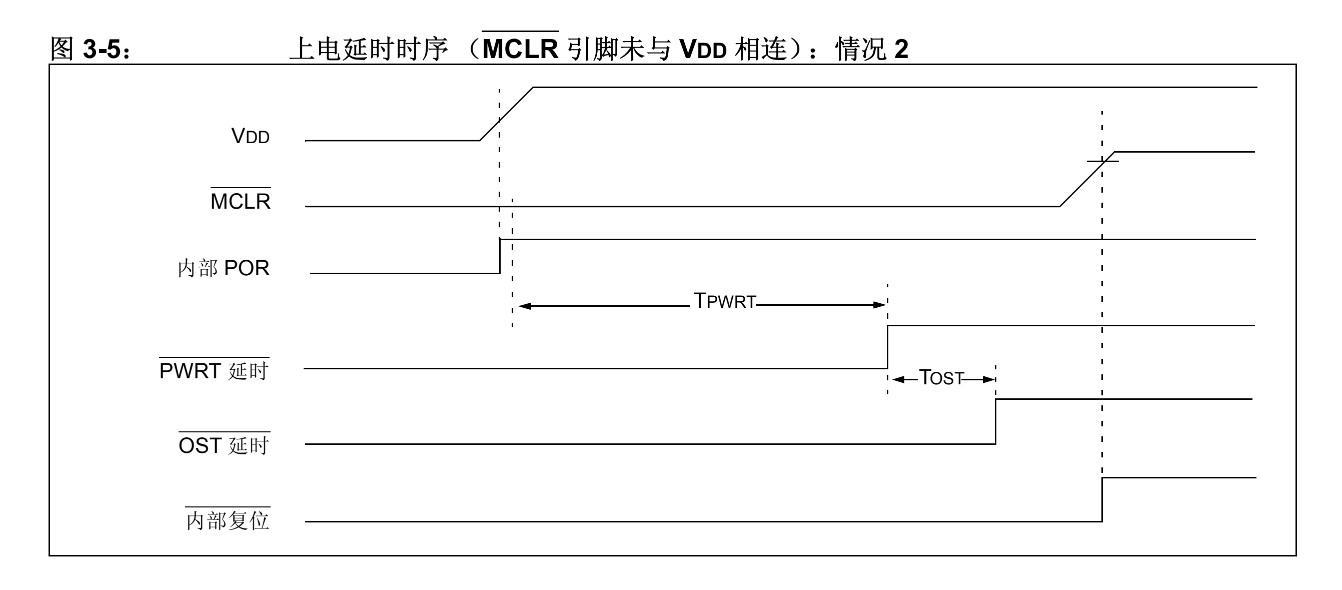


图 3-6： 缓慢上升时间 （$\overline{\text{MCLR}}$ 引脚与 VDD 相连）

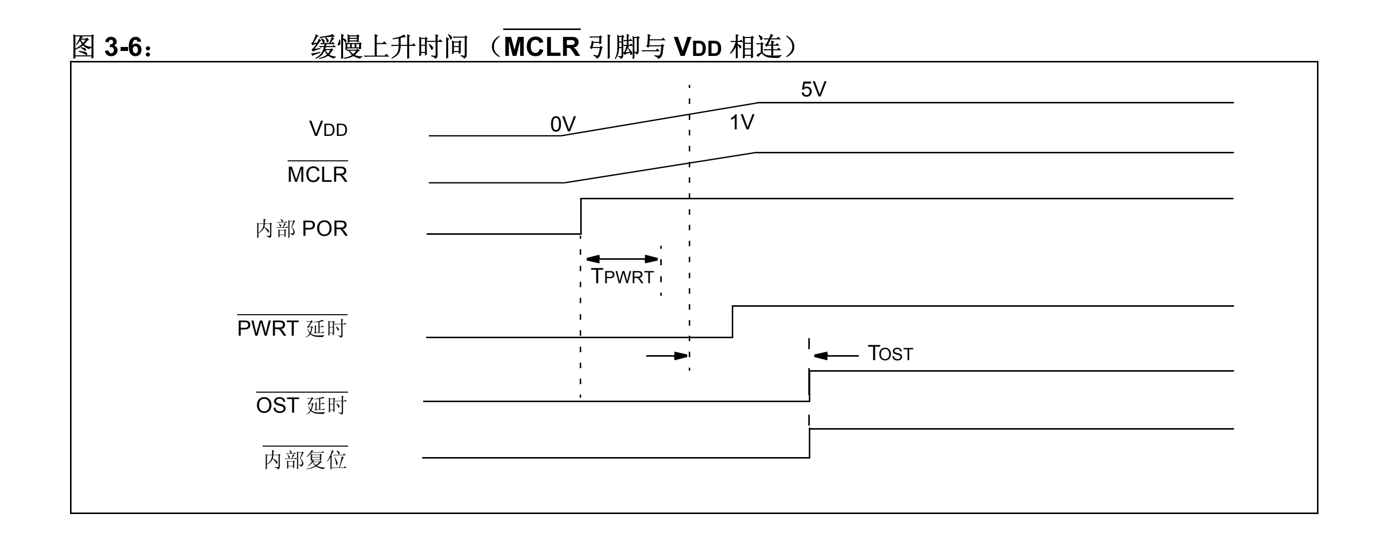
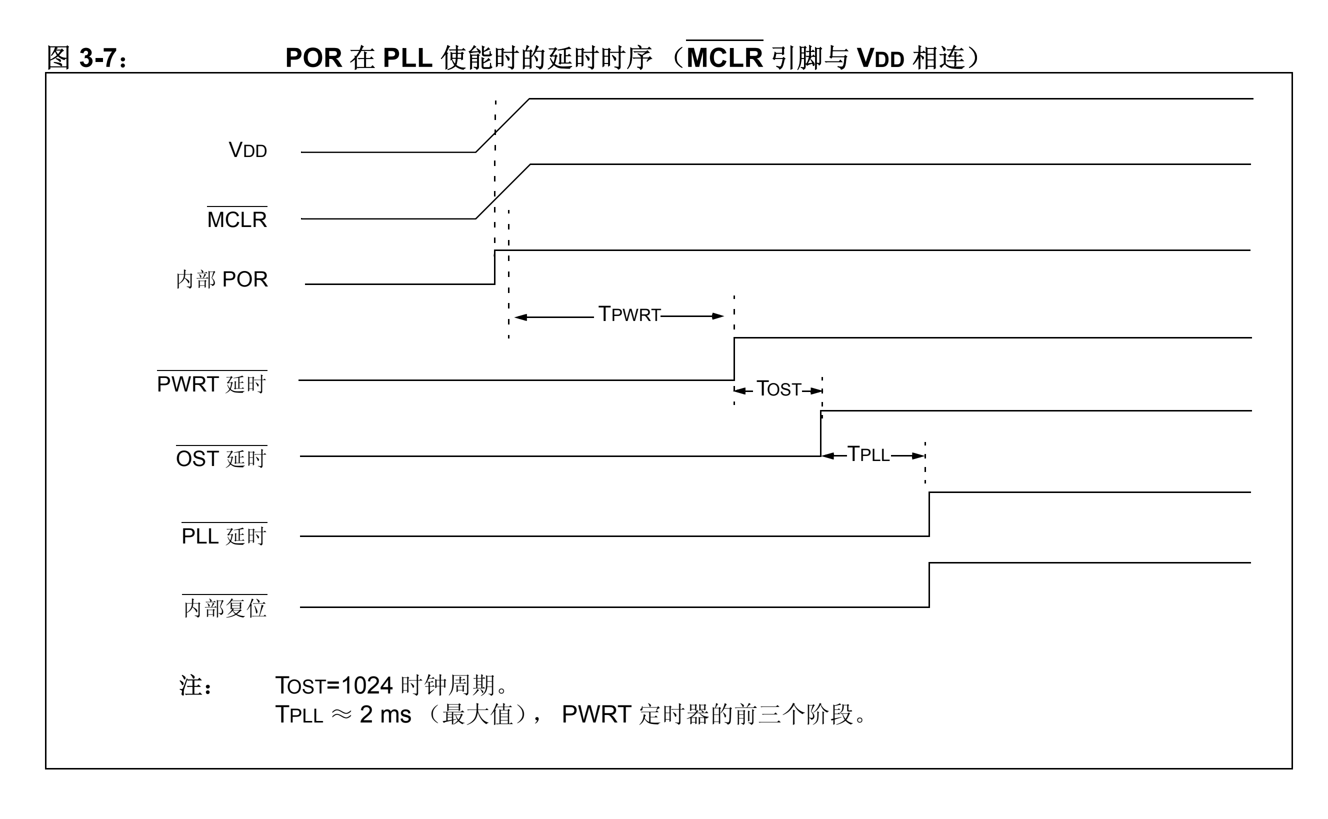


图 3-7： POR 在 PLL 使能时的延时时序 （$\overline{\text{MCLR}}$ 引脚与 VDD 相连）



注： TOST=1024 时钟周期。
TPLL ≈ 2 ms （最大值）， PWRT 定时器的前三个阶段。

注：

4.0 存储器构成

在增强型 MCU 器件中有三种存储器模块。这些存储器模块是：

- 程序存储器
- 数据 RAM
- 数据 EEPROM

数据和程序存储器各自使用不同的总线，这样就能同时访问这些模块。

第 5.0 节和第 6.0 节分别提供了闪存程序存储器和数据 EEPROM 的其他详细信息。

4.1 程序存储器构成

21 位的程序计数器可以寻址 2MB 的程序存储器空间。访问物理实现存储器和这个 2MB 地址之间的存储单元时，会导致读取都为 0 （NOP 指令）。

PIC18F252 和 PIC18F452 都带有 32 KB 的闪存存储器，而 PIC18F242 和 PIC18F442 则带有 16 KB 的闪存存储器。这就是说 PIC18FX52 最多可以存储 16K 的单字指令，而 PIC18FX42 则最多可以存储 8K 的单字指令。

复位向量地址为 0000h，中断向量地址为 0008h 和 0018h。

图 4-1 显示了 PIC18F242/442 器件的程序存储器映射，图 4-2 显示了 PIC18F252/452 器件的程序存储器映射。

图 4-1：PIC18F442/242 的程序存储器映射和堆栈

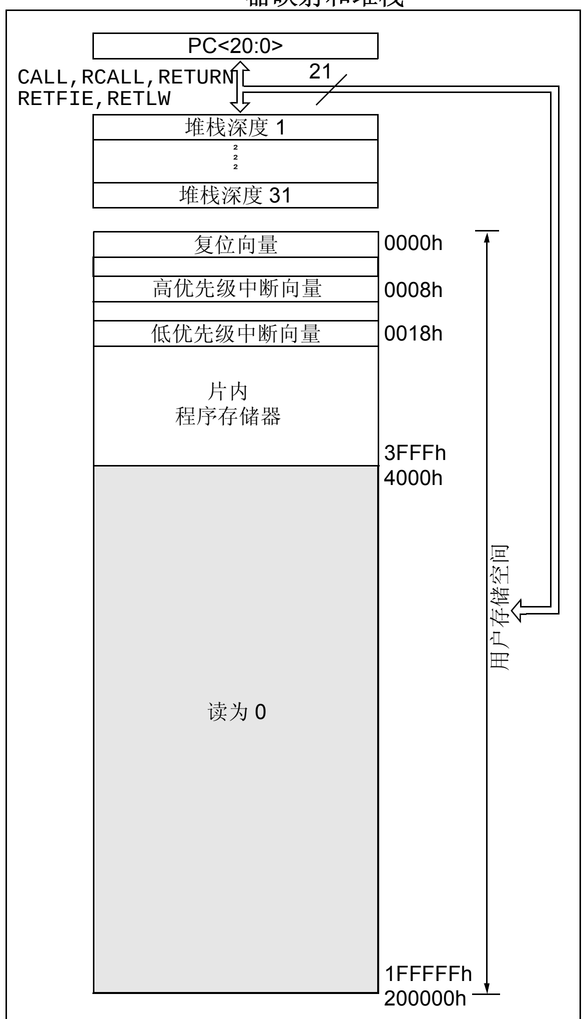
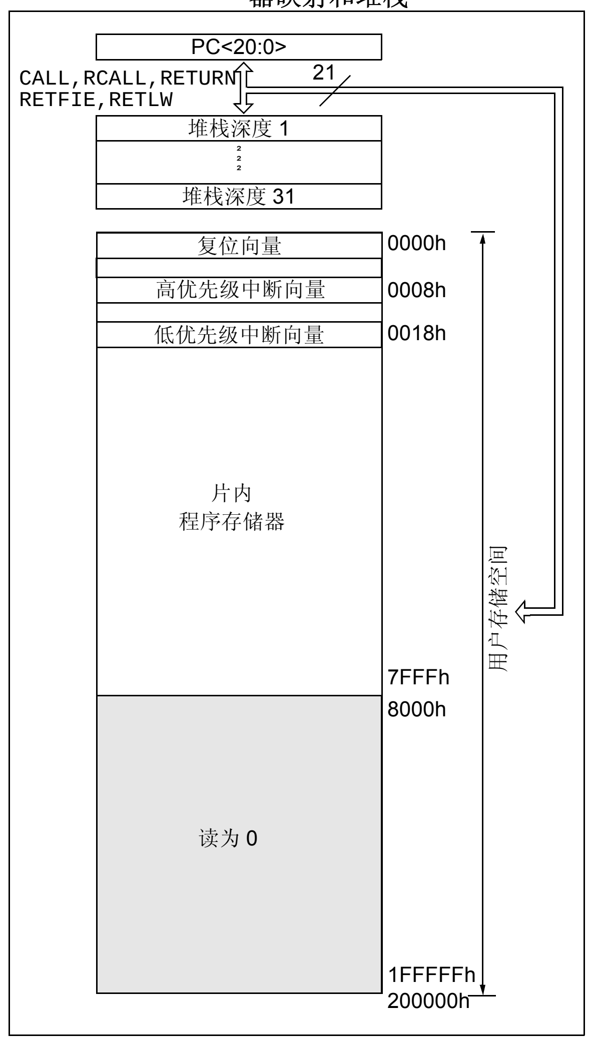


图 4-2：PIC18F452/252 的程序存储器映射和堆栈



4.2 返回地址堆栈

返回地址堆栈最多允许31个程序调用和中断的组合。当执行 CALL 或 RCALL 指令或发生中断时，程序计数器（Program Counter， PC）的值会被压入堆栈。而执行RETURN、RETLW或RETFIE指令时，PC值从堆栈弹出。PCLATU和PCLATH不受RETURN或CALL指令的影响。

堆栈通过 21 位的 RAM 和一个 5 位的栈指针以 31 个字来工作，其中栈指针在全部复位后会初始化为 00000b。栈指针 00000b 不与任何 RAM 关联。这只是复位值。执行 CALL 类型指令时，产生进栈操作，栈指针首先加 1，并且将 PC 的内容写入栈指针指向的 RAM 单元。执行 RETURN 类型指令时，产生出栈操作，STKPTR 所指向的 RAM 单元的内容会传递给 PC，并且栈指针减 1。

堆栈空间既不属于程序空间也不属于数据空间。栈指针可以读写，并且通过 SFR 寄存器可以读写栈顶地址。使用栈顶SFR还可以将数据压入或弹出堆栈。状态位表明栈指针是否位于所提供的 31 级，还是超出了该范围。

4.2.1 栈顶访问

栈顶是可读写的。三个寄存器单元 TOSU、 TOSH 和 TOSL 保存 STKPTR 寄存器所指向的堆栈单元的内容。这可以让用户在必要时实现软件堆栈。在CALL、RCALL 或中断后，软件可以读取 TOSU、TOSH 和 TOSL 寄存器来获取进栈值。这些值可以被置入用户定义的软件堆栈。返回时，软件可以替换 TOSU、TOSH 和 TOSL 的内容并执行返回。

为防止意外的堆栈操作，在此期间用户必须禁止全局中断使能位。

4.2.2 返回栈指针 （STKPTR）

STKPTR 寄存器包含栈指针值、STKFUL（堆栈满）状态位和 STKUNF（堆栈下溢）状态位。寄存器 4-1 显示了 STKPTR 寄存器。栈指针可以是 0 到 31 之间的任一个值。当向堆栈压入值时，栈指针加 1；而从堆栈弹出值时，栈指针减 1。在复位时，堆栈指针为 0。用户可以读写栈指针的值。实时操作系统可以利用此特性维护返回堆栈。

当向堆栈压入 PC 值 31 次（且没有从堆栈弹出任何值）后， STKFUL 位就会置 1。只能用软件或通过 POR 将 STKFUL 位清零。

根据堆栈溢出复位使能 STVREN （Stack Overflow Reset Enable）配置位的状态，当堆栈满时执行此操作。如需了解器件配置位的说明，请参阅第 20.0 节。如果 STVREN 置 1 （缺省值），第 31 次进栈将把 （PC + 2）值压入堆栈，将 STKFUL 位置 1，并复位器件。STKFUL 位将保持置 1，而栈指针将被置 0。

如果 STVREN 被清零，第 31 次进栈时 STKFUL 位会被置 1，栈指针则加 1 变为 31。任何其他进栈都不会覆盖第 31 次的进栈值，并且 STKPTR 将保持为 31。

当堆栈弹出的次数足以卸空堆栈时，下一次出栈会向 PC 返回一个零值，并将 STKUNF 位置 1，而栈指针则保持为 0。 STKUNF 位将保持置 1，直到软件清零或发生 POR。

注：	下溢引起的向PC返回零值会将程序指向复位向量，此时可以验证堆栈状态并采取相应的操作。

寄存器 **4-1**： **STKPTR 寄存器**

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKOVF	STKUNF	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

bit 7(1) **STKOVF**：堆栈满标志位
1= 栈满或栈溢出
0= 栈未满或未溢出

bit 6(1) **STKUNF**：堆栈下溢标志位
1 = 发生堆栈下溢
0= 没有发生堆栈下溢

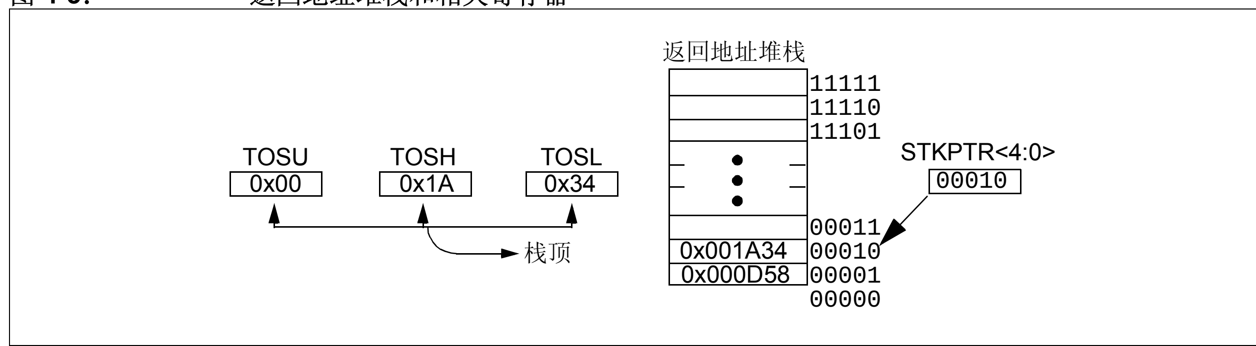
bit 5 **未实现位**：读为 0

bit 4-0 **SP4:SP0**：栈指针单元位

注 **1**：只能通过用户软件或 POR 将 bit 7 和 bit 6 清零。

图注：			
R= 可读位	W= 可写位	U= 未实现位，读为 0	
-n= 上电复位时的值	1= 置位	0= 清零	x= 未知位

图 **4-3**： 返回地址堆栈和相关寄存器



4.2.3 PUSH 和 POP 指令

因为栈顶（Top-of-Stack， TOS）可以读写，所以能够将值压入堆栈或从堆栈弹出而不影响程序的正常执行，这样会是一个理想情况。要将当前 PC 值压入堆栈，可以执行 PUSH 指令。这将使栈指针加 1，并将当前 PC 值装入堆栈。然后就可以修改 TOSU、 TOSH 和 TOSL，将返回地址放到堆栈中。

使用 POP 指令可以从堆栈中取出 TOS 值，并用前一个入栈值来更新 TOS 值，而不会影响程序的正常执行。POP 指令通过将栈指针减 1 来丢弃当前的 TOS。然后前一个入栈值就成为 TOS 值。

4.2.4 堆栈满 / 下溢复位

对 STVREN 配置位编程可以使能这些复位。如果禁止了 STVREN 位，堆栈满或堆栈下溢情况会将相应的 STKFUL 或 STKUNF 位置 1，但不会使器件复位。如果使能了 STVREN 位，堆栈满或堆栈下溢情况会将相应的 STKFUL 或 STKUNF 位置 1，然后使器件复位。只能通过用户软件或 POR 复位将 STKFUL 或 STKUNF 位清零。

4.3 快速寄存器堆栈

中断可以使用“快速中断返回”选项。为 STATUS、WREG 和 BSR 寄存器提供了快速寄存器堆栈，其深度仅为 1。此堆栈不可读写。当处理器用于中断时，堆栈装入对应寄存器的当前值。如果使用 FAST RETURN 指令从中断返回，这些寄存器中的值就会装回工作寄存器。

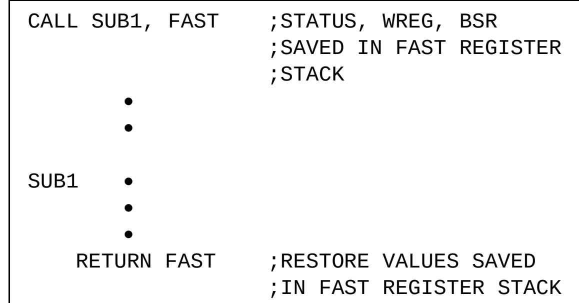
低优先级或高优先级中断源会将值压入堆栈寄存器。如果同时使能了低优先级中断和高优先级中断，低优先级中断将无法可靠使用堆栈寄存器。如果在为低优先级中断提供服务时，发生了高优先级中断，则低优先级中断存储的堆栈寄存器值将被覆盖。

如果在使用低优先级中断时没有禁止高优先级中断，用户必须在低优先级中断期间用软件保存关键寄存器。

如果没有使用中断，快速寄存器堆栈可以用于在子程序调用结束时恢复 STATUS、WRE 和 BSR 寄存器。要在子程序调用中使用快速寄存器堆栈，必须执行 FAST CALL 指令。

例 4-1 显示了使用快速寄存器堆栈的源代码示例。

例 4-1： 快速寄存器堆栈代码示例



```
CALL SUB1, FAST     ;STATUS, WREG, BSR
                    ;SAVED IN FAST REGISTER
                    ;STACK
        •
        •

SUB1    •
        •
        •
    RETURN FAST     ;RESTORE VALUES SAVED
                    ;IN FAST REGISTER STACK
```

4.4 PCL、 PCLATH 和 PCLATU

程序计数器（PC）指定要取出执行的指令地址。PC 的宽度为 21 位。其中的低字节称为 PCL 寄存器，该寄存器可读写的；高字节称为 PCH 寄存器。该寄存器包含 PC<15:8> 位，不能直接读写。可以通过 PCLATH 寄存器更新 PCH 寄存器。更高字节称为 PCU。该寄存器包含 PC<20:16> 位，不能直接读写。可以通过 PCLATU 寄存器更新 PCU 寄存器。

PC 寻址程序存储器中的字节。为防止 PC 偏离字指令，PCL 的 LSB 固定为 0 值。PC 在程序存储器中以 2 为增量对连续指令寻址。

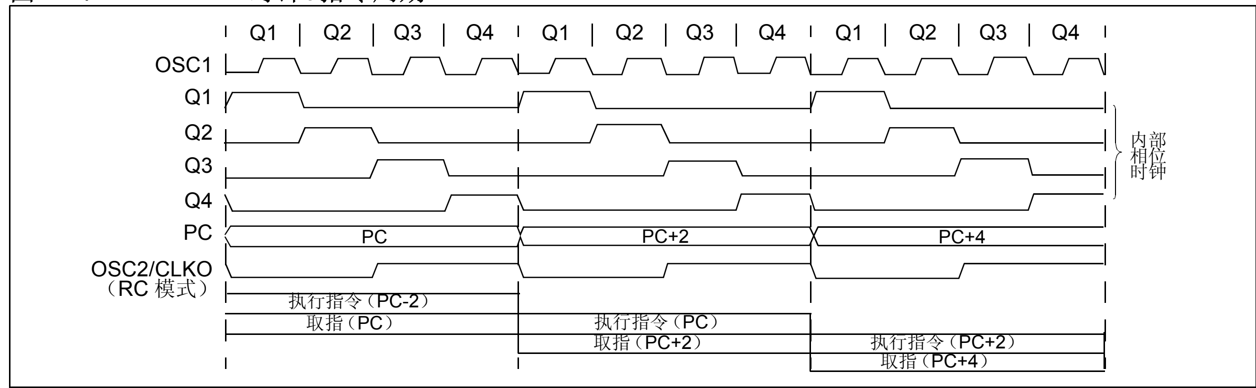
CALL、RCALL、GOTO 和程序转移指令直接写入程序计数器。对于这些指令，PCLATH 和 PCLATU 的内容不会传递到程序计数器。

通过写 PCL 的操作，PCLATH 和 PCLATU 的内容将被传输到程序计数器。类似地，通过读 PCL 的操作，程序计数器的高两位字节将被传送到PCLATH 和PCLATU。这有助于计算 PC 的偏移量 （请参见第 4.8.1 节）。

4.5 时序图 / 指令周期

由 OSC1 引脚输入的时钟信号在器件内部经过 4 分频后产生 4 个不重叠的正交时钟信号，即 Q1、 Q2、 Q3 和 Q4。程序计数器 （PC）在每个 Q1 加 1，并从程序存储器取指令，在 Q4 将指令锁存到指令寄存器中。指令的译码和执行在下一个 Q1 到 Q4 中完成。图 4-4 所示为时钟和指令执行流图。

图 4-4： 时钟 / 指令周期



4.6 指令流 / 流水线

一个指令周期由 4 个 Q 周期组成（即 Q1、Q2、Q3 和 Q4）。取指和指令执行是流水执行的，用一个指令周期来取指，而用另一个指令周期来译码和执行。但是由于是流水线操作，每条指令的等效执行时间为一个指令周期。如果某条指令改变了程序计数器（如 GOTO 指令），则需要两个指令周期才能完成该指令 （见例 4-2）。

在 Q1 周期，开始取指周期，程序计数器（PC）以 1 递增。

在执行周期，所取指令在 Q1 周期中被锁存到指令寄存器 （Instruction Register， IR）。在随后的 Q2、Q3 和 Q4 周期中译码并执行该指令。其中在 Q2 周期读取数据存储器 （读操作数），在 Q4 周期写数据存储器 （写目标）。

例 4-2： 指令流水线流程

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOVLW 55h	取指 1	执行 1				
2. MOVWF PORTB		取指 2	执行 2			
3. BRA SUB_1			取指 3	执行 3		
4. BSF PORTA, BIT3 (Forced NOP)				取指 4	清空 （NOP）	
5. Instruction @ address SUB_1					取指 SUB_1	执行 SUB_1

除程序转移指令外，所有的指令都是单周期指令。由于在取新指和指令执行的同时要从流水线清空取指，所以程序转移指令需要两个周期。

4.7 程序存储器中的指令

程序存储器是按字节寻址的。指令以 2 字节或 4 字节形式存储在程序存储器中。指令字的低位字节始终存储在偶数地址的程序存储单元中 （LSB='0'）。图 4-5 举例说明了指令字在程序存储器中的保存方式。要保持与指令边界对齐，PC 以 2 为单位递增，并且 LSB 总是读为 0 （见第 4.4 节）。

CALL 和 GOTO 指令在指令中嵌入了程序存储器的绝对地址。指令总是存储在字边界，因而指令所包含的地址为字地址。字地址会写入 PC<20:1>，后者则在程序存储器中访问所需的字节地址。图 4-5 中的指令 2 说明了指令 “GOTO 000006h” 在程序存储器中是如何进行编码的。程序转移指令也采取同样的方式对相对地址偏移量进行编码。在转移指令中的偏移值代表单字指令数，PC 将以此作为偏移量。第 20.0节提供了有关指令集的更详细信息。

图 4-5： 程序存储器中的指令

程序存储器字节单元 →

指令			LSB=1	LSB=0	字地址 ↓
					000000h
					000002h
					000004h
					000006h
指令 1：	MOVLW	055h	0Fh	55h	000008h
指令 2：	GOTO	000006h	EFh	03h	00000Ah
			F0h	00h	00000Ch
指令 3：	MOVFF	123h, 456h	C1h	23h	00000Eh
			F4h	56h	000010h
					000012h
					000014h

4.7.1 双字指令

PIC18FXX2 器件有 4 个双字指令：MOVFF、CALL、GOTO和LFSR。这些指令第二个字的4个MSB（最高位）均置为 1，是一种特殊的 NOP 指令。第二个字的低 12 位包含指令要使用的数据。如果执行了指令的第一个字，就会访问第二个字中的数据。如果自行执行指令的第二个字（跳过了第一个字），则其效果将相当于一个NOP指令。如果双字指令跟在修改PC的条件指令后，就有必要执行此操作。例 4-3 中的程序示例说明了这一概念。如需了解有关此指令集的更多信息，请参阅第 20.0 节。

示例 4-3： 双字指令

情形 1：	
目标代码	源代码
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2 ; No, execute 2-word instruction
1111 0100 0101 0110	; 2nd operand holds address of REG2
0010 0100 0000 0000	ADDWF REG3 ; continue code

情形 2：	
目标代码	源代码
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2 ; Yes
1111 0100 0101 0110	; 2nd operand becomes NOP
0010 0100 0000 0000	ADDWF REG3 ; continue code

4.8 查找表

有两种方式可实现查找表，它们是：

- 计算 GOTO
- 表读取

4.8.1 计算 GOTO

计算 GOTO 是通过向程序计数器加一个偏移量来实现的（ADDWF PCL）。

使用 ADDWF PCL 指令和一组 RETLW 0xnn 指令可组成一个查找表。在调用该表前，会先将偏移值装入 WREG 中。被调用程序的第一条指令是 ADDWF PCL 指令。接下去执行的是 RETLW 0xnn 指令，它将向调用函数返回值 0xnn。

偏移值（WREG 中的值）指定程序计数器应该增加的字节数。

在这种方式中，每个指令单元只能存储一个数据字节，并且需要返回地址堆栈还有空余空间。

注：	ADDWF PCL 指令不会更新 PCLATH 和 PCLATU。要更新 PCLATH 和 PCLATU 必须执行对 PCL 的读取操作。

4.8.2 表读取 / 表写入

有一种更好方法可以将数据存储在程序存储器，可以在每个指令单元存储 2 个字节的数据。

运用表读取和表写入，每个程序字可以存储 2 个字节的查找表数据。表指针（TBLPTR）指定字节地址，而表锁存器（TABLAT）则包含从程序存储器读取或写入程序存储器的数据。进出程序存储器的数据每次为一个字节。

第 5.0 节描写了表读取 / 表写入操作。

4.9 数据存储器构成

数据存储器作为静态RAM形式实现。在数据存储器中，每个寄存器有 12 位地址，数据存储器可达 4096 个字节。图 4-6 和图 4-7 显示了 PIC18FXX2 器件数据存储器的构成。

数据存储器映射分为 16 个存储区，每个存储区有 256 字节。存储区选择寄存器（Bank Select Register，BSR）的低 4 位（BSR<3:0>）选择将要访问的存储区。BSR 的高 4 位没有使用。

数据存储器由特殊功能寄存器（Special Function Register，SFR）和通用寄存器（General Purpose Register，GPR）组成。SFR 用于控制器和外设模块的控制和状态显示，GPR则用于在用户应用程序中存储数据和改写操作。SFR 从存储区 15 的最末单元（0xFFF）开始并向下扩展。存储区中SFR以外的所有其余空间都可以作为 GPR。 GPR 从存储区 0 的第一个单元开始，并向上分布。读取任何未使用单元时，将读为 0。

整个数据存储器可以采用直接寻址或间接寻址来访问。直接寻址可能需要使用 BSR 寄存器。间接寻址需要使用文件选择寄存器（File Select Register，FSRn）和相应的间接文件操作数（Indirect File Operand，INDFn）。每个 FSR 保存一个 12 位的地址值，使用该值无需选择存储区即可访问数据存储器映射的任何单元。

这样的指令集和架构支持跨所有存储区的操作。这可以通过间接寻址或使用 MOVFF 指令实现。 MOVFF 指令是一个双字 / 双周期指令，它将值从一个寄存器移到另一个寄存器。

无论当前 BSR 值如何，要确保能在一个周期存取常用寄存器（SFR 和所选的 GPR），需要实现一个快速访问存储区（Access Bank）。存储区 0 的一段和存储区 15 的一段构成了存取 RAM（Access RAM）。第 4.10 节详细介绍了存取 RAM。

4.9.1 通用寄存器文件

可以采用直接寻址或间接寻址来访问存储器文件。间接寻址使用文件选择寄存器和相应的间接文件操作数进行。第 4.12 节说明了间接寻址的过程。

增强型MCU器件可能将GPR区分成不同的存储区。上电复位不会初始化GPR，并且其他复位也不会改变其内容。

所有指令都可以使用数据RAM作为GPR寄存器。存储区 15 的前半部分（0xF80 到 0xFFF）包含 SFR。数据存储器的所有其他存储区从存储区 0 开始包含 GPR 寄存器。

4.9.2 特殊功能寄存器

特殊功能寄存器（SFR）是 CPU 和外设模块用来控制器件操作的寄存器。这类寄存器作为静态 RAM 形式实现。表 4-1 和表 4-2 列出了这些寄存器。

SFR 可分为两类，一类与内核功能模块有关，另一类与外设功能模块有关。本节将讲述与内核功能有关的特殊功能寄存器，另一类与外设功能操作有关的特殊功能寄存器将在相应的外设功能模块章节中论述。

SFR 通常分布在外设中，用来控制外设功能。

未使用的 SFR 单元将不会实现，并读为 0。有关 SFR 的地址信息，请参见表 4-1。

图 4-6： PIC18F242/442 的数据存储器映射

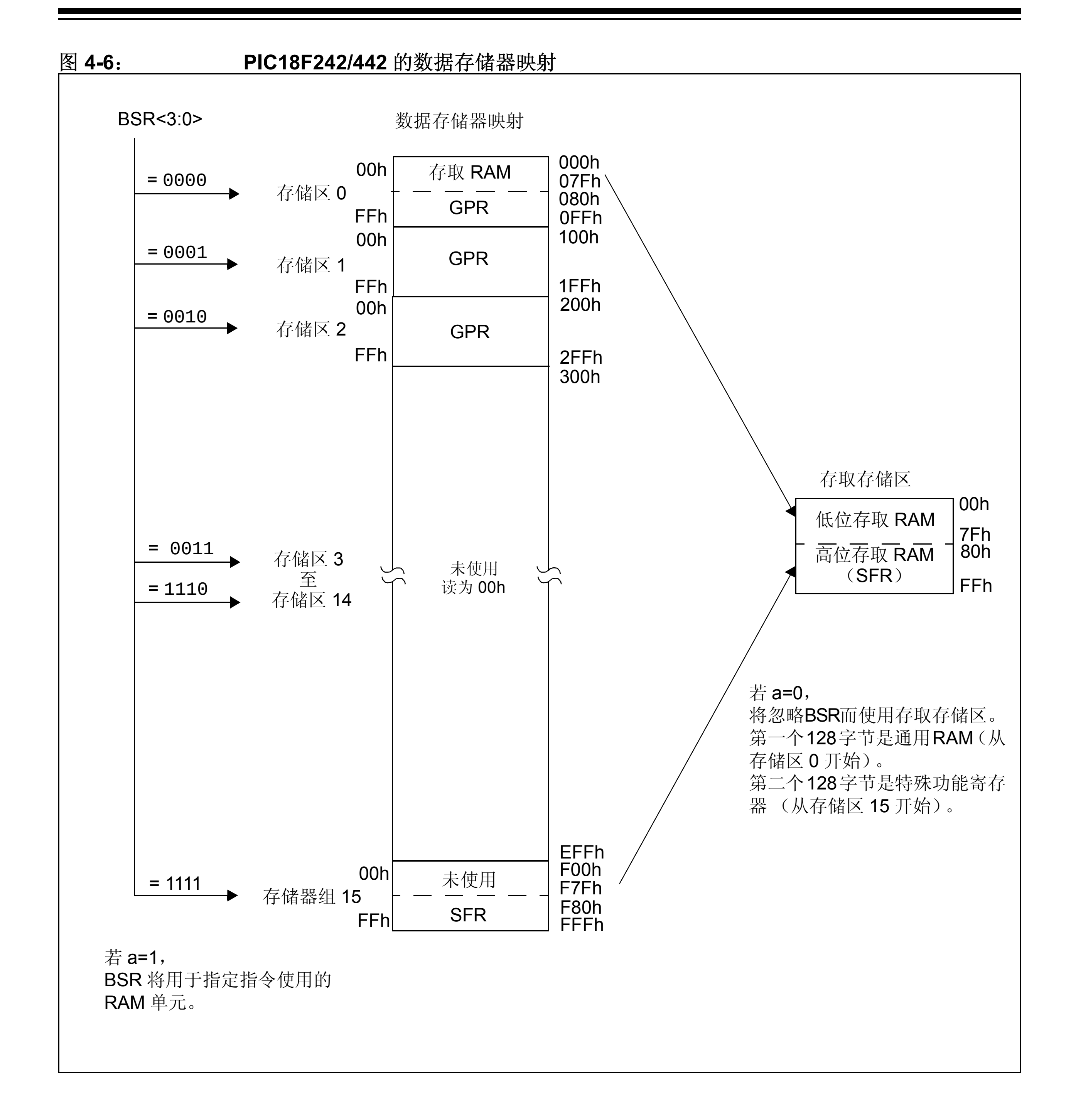


图 4-7： PIC18F252/452 的数据存储器映射

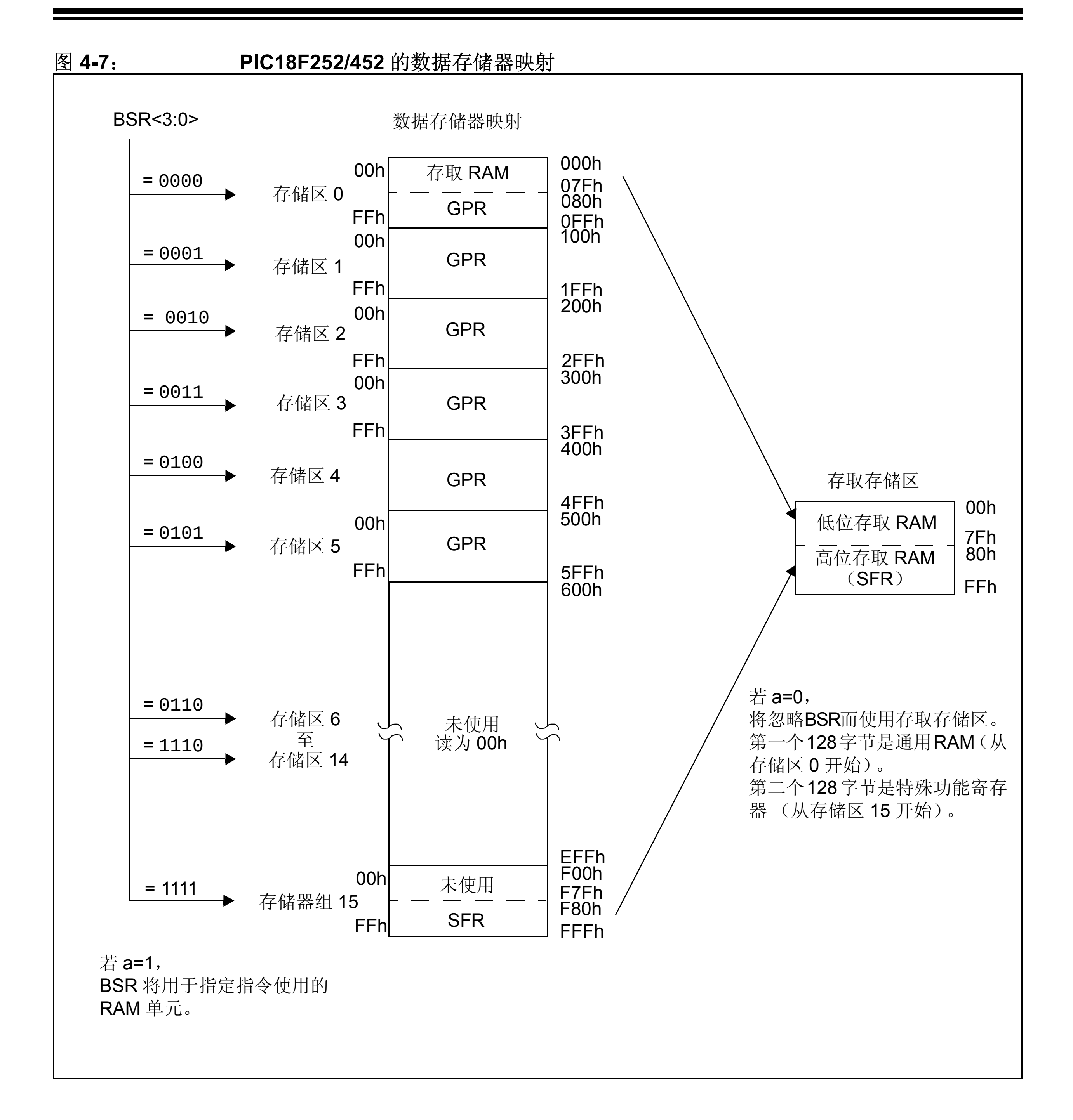


表 **4-1**： 特殊功能寄存器映射

地址	名称	地址	名称	地址	名称	地址	名称
FFFh	TOSU	FDFh	INDF2[3]	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2[3]	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2[3]	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2[3]	FBCh	CCPR2H	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2[3]	FBBh	CCPR2L	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	—
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	—
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	—
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—	F97h	—
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—	F96h	TRISE[2]
FF5h	TABLAT	FD5h	T0CON	FB5h	—	F95h	TRISD[2]
FF4h	PRODH	FD4h	—	FB4h	—	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	—
FF0h	INTCON3	FD0h	RCON	FB0h	—	F90h	—
FEFh	INDF0[3]	FCFh	TMR1H	FAFh	SPBRG	F8Fh	—
FEEh	POSTINC0[3]	FCEh	TMR1L	FAEh	RCREG	F8Eh	—
FEDh	POSTDEC0[3]	FCDh	T1CON	FADh	TXREG	F8Dh	LATE[2]
FECh	PREINC0[3]	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD[2]
FEBh	PLUSW0[3]	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	—
FE7h	INDF1[3]	FC7h	SSPSTAT	FA7h	EECON2	F87h	—
FE6h	POSTINC1[3]	FC6h	SSPCON1	FA6h	EECON1	F86h	—
FE5h	POSTDEC1[3]	FC5h	SSPCON2	FA5h	—	F80h	—
FE4h	PREINC1[3]	FC4h	ADRESH	FA4h	—	F84h	PORTE[2]
FE3h	PLUSW1[3]	FC3h	ADRESL	FA3h	—	F83h	PORTD[2]
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	—	FA0h	PIE2	F80h	PORTA

注 **1**：未实现的寄存器读为 0。
2：PIC18F2X2 器件没有此寄存器。
3：这不是物理寄存器。

表 **4-2**： 寄存器文件小结

文件名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	详情请见（页码）：
TOSU	—	—	—	栈顶高位字节 （TOS<20:16>）					---0 0000	37
TOSH	栈顶高位字节 （TOS<15:8>）								0000 0000	37
TOSL	栈顶低位字节 （TOS<7:0>）								0000 0000	37
STKPTR	STKFUL	STKUNF	—	返回栈指针					00-0 0000	38
PCLATU	—	—	—	PC<20:16> 的保持寄存器					---0 0000	39
PCLATH	PC<15:8> 的保持寄存器								0000 0000	39
PCL	PC 低位字节 （PC<7:0>）								0000 0000	39
TBLPTRU	—	—	bit21[2]	程序存储器表指针高位字节 （TBLPTR<20:16>）					--00 0000	58
TBLPTRH	程序存储器表指针高位字节 （TBLPTR<15:8>）								0000 0000	58
TBLPTRL	程序存储器表指针低位字节 （TBLPTR<7:0>）								0000 0000	58
TABLAT	程序存储器表锁存器								0000 0000	58
PRODH	乘积寄存器高位字节								xxxx xxxx	71
PRODL	乘积寄存器低位字节								xxxx xxxx	71
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	75
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1	76
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	77
INDF0	使用 FSR0 的内容寻址数据存储器， FSR0 的值不改变 （不是物理寄存器）								n/a	50
POSTINC0	使用 FSR0 的内容寻址数据存储器， FSR0 的值会寻址后递增 （不是物理寄存器）								n/a	50
POSTDEC0	使用 FSR0 的内容寻址数据存储器， FSR0 的值会寻址后递减 （不是物理寄存器）								n/a	50
PREINC0	使用 FSR0 的内容寻址数据存储器， FSR0 的值会寻址前递增 （不是物理寄存器）								n/a	50
PLUSW0	使用 FSR0 的内容寻址数据存储器， FSR0 的值 （不是物理寄存器）。按 WREG 中的值偏移。								n/a	50
FSR0H	—	—	—	—	间接数据存储器地址指针 0 的高位字节				---- 0000	50
FSR0L	间接数据存储器地址指针 0 的低位字节								xxxx xxxx	50
WREG	工作寄存器								xxxx xxxx	n/a
INDF1	使用 FSR1 的内容寻址数据存储器， FSR1 的值不改变 （不是物理寄存器）								n/a	50
POSTINC1	使用 FSR1 的内容寻址数据存储器， FSR1 的值会寻址后递增 （不是物理寄存器）								n/a	50
POSTDEC1	使用 FSR1 的内容寻址数据存储器， FSR1 的值会寻址后递减 （不是物理寄存器）								n/a	50
PREINC1	使用 FSR1 的内容寻址数据存储器， FSR1 的值会寻址前递增 （不是物理寄存器）								n/a	50
PLUSW1	使用 FSR1 的内容寻址数据存储器， FSR1 的值 （不是物理寄存器）。按 WREG 中的值偏移。								n/a	50
FSR1H	—	—	—	—	间接数据存储器地址指针 1 的高位字节				---- 0000	50
FSR1L	间接数据存储器地址指针 1 的低位字节								xxxx xxxx	50
BSR	—	—	—	—	存储区选择寄存器				---- 0000	49
INDF2	使用 FSR2 的内容寻址数据存储器， FSR2 的值不改变 （不是物理寄存器）								n/a	50
POSTINC2	使用 FSR2 的内容寻址数据存储器， FSR2 的值会寻址后递增 （不是物理寄存器）								n/a	50
POSTDEC2	使用 FSR2 的内容寻址数据存储器， FSR2 的值会寻址后递减 （不是物理寄存器）								n/a	50
PREINC2	使用 FSR2 的内容寻址数据存储器， FSR2 的值会寻址前递增 （不是物理寄存器）								n/a	50
PLUSW2	使用 FSR2 的内容寻址数据存储器， FSR2 的值 （不是物理寄存器）。按 WREG 中的值偏移。								n/a	50
FSR2H	—	—	—	—	间接数据存储器地址指针 2 的高位字节				---- 0000	50
FSR2L	间接数据存储器地址指针 2 的低位字节								xxxx xxxx	50
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	52
TMR0H	Timer0 寄存器高位字节								0000 0000	105
TMR0L	Timer0 寄存器低位字节								xxxx xxxx	105
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	103

图注： x= 未知， u = 未改变， -= 未实现， q= 值随条件而变化

注 **1**： 仅在 RCIO 和 ECIO 振荡器模式下， RA6 和相关位被配置为端口引脚，在所有其他振荡器模式下均读为 0。

2： TBLPTRU 的 Bit 21 允许对器件配置位进行存取。

3： PIC18F2X2 器件中保留了这些寄存器和位，应保持其清零状态。

表 **4-2**： 寄存器文件小结 （续）

文件名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	详情请见（页码）：
OSCCON	—	—	—	—	—	—	—	SCS	---- ---0	21
LVDCON	—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0	--00 0101	191
WDTCON	—	—	—	—	—	—	—	SWDTE	---- ---0	203
RCON	IPEN	—	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	0--1 11qq	53, 27, 84
TMR1H	Timer1 寄存器高位字节								xxxx xxxx	107
TMR1L	Timer1 寄存器低位字节								xxxx xxxx	107
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	107
TMR2	Timer2 寄存器								0000 0000	111
PR2	Timer2 周期寄存器								1111 1111	112
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	111
SSPBUF	SSP 接收缓冲器 / 发送寄存器								xxxx xxxx	125
SSPADD	I^2C 从模式下的 SSP 地址寄存器。 I^2C 主模式下的 SSP 波特率重载寄存器。								0000 0000	134
SSPSTAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF	0000 0000	126
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	127
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	137
ADRESH	A/D 结果寄存器高位字节								xxxx xxxx	187,188
ADRESL	A/D 结果寄存器低位字节								xxxx xxxx	187,188
ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON	0000 00-0	181
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	182
CCPR1H	捕捉 / 比较 /PWM 寄存器 1 高位字节								xxxx xxxx	121, 123
CCPR1L	捕捉 / 比较 /PWM 寄存器 1 低位字节								xxxx xxxx	121, 123
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	117
CCPR2H	捕捉 / 比较 /PWM 寄存器 2 高位字节								xxxx xxxx	121, 123
CCPR2L	捕捉 / 比较 /PWM 寄存器 2 低位字节								xxxx xxxx	121, 123
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	117
TMR3H	Timer3 寄存器高位字节								xxxx xxxx	113
TMR3L	Timer3 寄存器低位字节								xxxx xxxx	113
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0000 0000	113
SPBRG	USART1 波特率发生器								0000 0000	168
RCREG	USART1 接收寄存器								0000 0000	175, 178, 180
TXREG	USART1 发送寄存器								0000 0000	173, 176, 179
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	166
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	167
EEADR	数据 EEPROM 地址寄存器								0000 0000	65, 69
EEDATA	数据 EEPROM 数据寄存器								0000 0000	69
EECON2	数据 EEPROM 控制寄存器 2 （不是物理寄存器）								---- ----	65, 69
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	66

图注： x= 未知， u = 未改变， -= 未实现， q= 值随条件而变化

注 **1**： 仅在 RCIO 和 ECIO 振荡器模式下， RA6 和相关位被配置为端口引脚，在所有其他振荡器模式下均读为 0。

2： TBLPTRU 的 Bit 21 允许对器件配置位进行存取。

3： PIC18F2X2 器件中保留了这些寄存器和位，应保持其清零状态。

表 **4-2**： 寄存器文件小结 （续）

文件名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	详情请见（页码）：
IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	83
PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	79
PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	81
IPR1	PSPIP(3)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	82
PIR1	PSPIF(3)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	78
PIE1	PSPIE(3)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	80
TRISE(3)	IBF	OBF	IBOV	PSPMODE	—	PORTE 的数据方向位			0000 -111	98
TRISD(3)	PORTD 的数据方向控制寄存器								1111 1111	96
TRISC	PORTC 的数据方向控制寄存器								1111 1111	93
TRISB	PORTB 的数据方向控制寄存器								1111 1111	90
TRISA	—	TRISA6(1)	PORTA 的数据方向控制寄存器						-111 1111	87
LATE(3)	—	—	—	—	—	读 PORTE 数据锁存器，写 PORTE 数据锁存器			---- -xxx	99
LATD(3)	读 PORTD 数据锁存器，写 PORTD 数据锁存器								xxxx xxxx	95
LATC	读 PORTC 数据锁存器，写 PORTC 数据锁存器								xxxx xxxx	93
LATB	读 PORTB 数据锁存器，写 PORTB 数据锁存器								xxxx xxxx	90
LATA	—	LATA6(1)	读 PORTA 数据锁存器，写 PORTA 数据锁存器 (1)						-xxx xxxx	87
PORTE(3)	读 PORTE 引脚，写 PORTE 数据锁存器								---- -000	99
PORTD(3)	读 PORTD 引脚，写 PORTD 数据锁存器								xxxx xxxx	95
PORTC	读 PORTC 引脚，写 PORTC 数据锁存器								xxxx xxxx	93
PORTB	读 PORTB 引脚，写 PORTB 数据锁存器								xxxx xxxx	90
PORTA	—	RA6(1)	读 PORTA 引脚，写 PORTA 数据锁存器 (1)						-x0x 0000	87

图注： x= 未知， u = 未改变， -= 未实现， q= 值随条件而变化

注 **1**： 仅在 RCIO 和 ECIO 振荡器模式下， RA6 和相关位被配置为端口引脚，在所有其他振荡器模式下均读为 0。

2： TBLPTRU 的 Bit 21 允许对器件配置位进行存取。

3： PIC18F2X2 器件中保留了这些寄存器和位，应保持其清零状态。

4.10 存取存储区

存取存储区是一个增强架构，对于 C 编译器的代码优化非常有用。C 编译器采用的技术对于汇编语言编写的程序也可能会有用。

此数据存储区域可用于：

- 中间计算值
- 子程序的本地变量
- 变量的快速上下文保存 / 切换
- 常用变量
- SFR 的快速求值 / 控制 （无存储区选择）

存取存储区由存储区 15 （SFR）的高 128 字节和存储区 0 的低 128 字节组成。这两部分分别称为存取 RAM 高位和存取 RAM 低位。图 4-6 和图 4-7 指出了存取 RAM 区。

指令字中的 1 位指定操作应该在 BSR 寄存器指定的存储区发生还是在存取存储区中进行。此位表示为 a 位（表示存取位）。

当强制在存取存储区进行 （a=0）时，存取 RAM 低位中的末位地址后面是存取 RAM 高位的首地址。存取 RAM 高位会映射特殊功能寄存器，这样无需任何软件开销即可存取这些寄存器。这对于测试状态标志位和修改控制位很有用。

4.11 存储区选择寄存器 （BSR）

由于需要很大的通用存储空间，便产生了 RAM 存储区选择方案。数据存储器分为 16 个存储区。当使用直接寻址时，应该配置 BSR 来指定目标存储区。

BSR<3:0> 保存 12 位 RAM 地址的高 4 位。BSR<7:4> 位总是为 0，写入值不会有任何影响。

在指令集中提供了 MOVLB 指令以帮助选择存储器组。

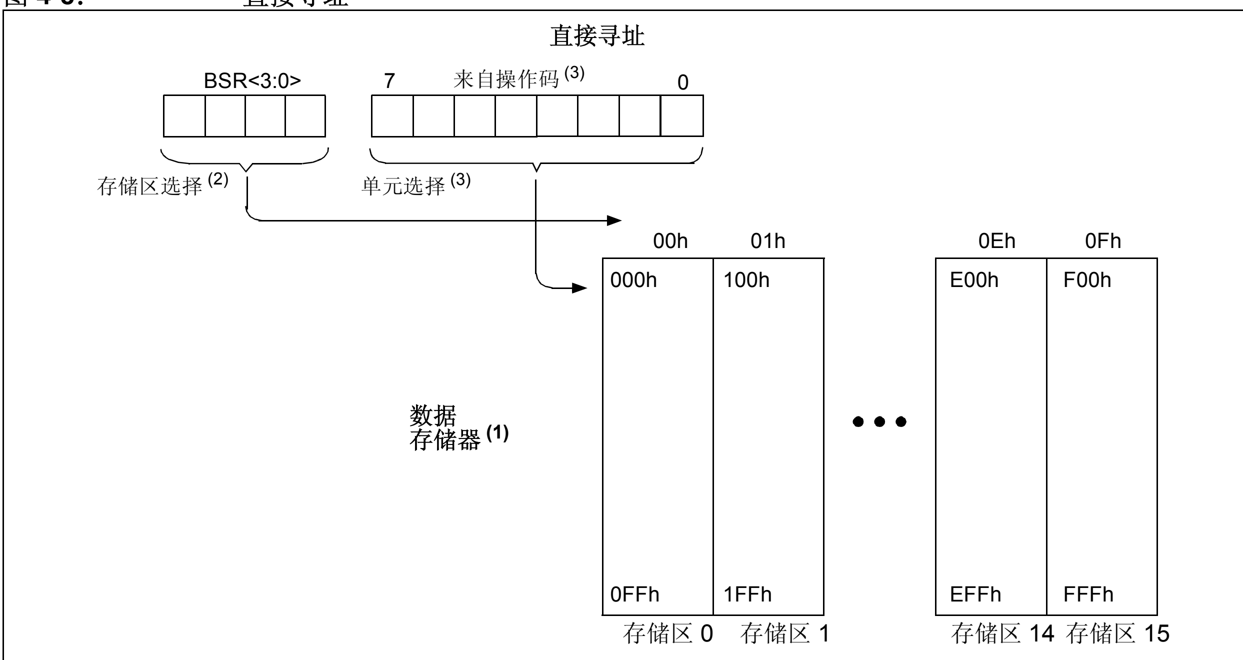
如果当前选择的存储区没有实现，任何读操作都会返回 0，写操作则被忽略。根据执行的指令， STATUS 寄存器位会相应被置 1 或清零。

每个存储区空间最大为 FFh （256 字节）。所有数据存储器都作为静态 RAM 实现。

因为 MOVFF 指令字嵌入了 12 位地址，所以该指令会忽略 BSR。

第 4.12 节说明了间接寻址，间接寻址可以对整个 RAM 空间进行线性寻址。

图 4-8： 直接寻址



注 1：如需了解寄存器文件映射的具体信息，请参阅表 4-1。
2：可以用指令的存取位强制选定的存储区 （BSR<3:0> 指定的）覆盖存取存储区的寄存器。
3：MOVFF 指令嵌入了完整的 12 位指令地址。

4.12 间接寻址， INDF 和 FSR 寄存器

间接寻址是对数据存储器寻址的一种模式，其中指令中的数据存储器地址不固定。FSR 寄存器充当指向要读写的数据存储器单元的指针。由于该指针位于 RAM 中，所以其内容可以通过程序修改。这对于数据存储器中的数据表和软件堆栈是有用的。图 4-9 是间接寻址工作情况，它实现了将数据值传送到由FSR寄存器的值指定的数据存储单元。

使用一个 INDF 寄存器就可能实现间接寻址。任何使用 INDF 寄存器的指令实际上访问的是由文件选择寄存器（FSR）指向的寄存器。如使用间接寻址方式（FSR=0）对 INDF 寄存器本身进行读操作，读出值将为 00h。而使用间接寻址对 INDF 寄存器进行写操作，实际执行的是空操作。如图 4-10 所示， FSR 寄存器包含一个 12 位地址。

INDFn 寄存器不是物理寄存器。对 INDFn 的寻址实际上是对 FSRn 寄存器所指之地址的寄存器寻址 （FSRn 是一个指针）。这就是间接寻址。

例 4-4 给出一个简单的示例，它用最少的指令说明怎样用间接寻址对存储区 1 （位置 100h-1FFh）中 RAM 单元进行清零。

例 4-4：　使用间接寻址将 RAM （存储区 1）清零的方法

```
        LFSR  FSR0 ,0x100 ;
NEXT    CLRF  POSTINC0     ; Clear INDF
                           ; register and
                           ; inc pointer
        BTFSS FSR0H, 1     ; All done with
                           ; Bank1?
        GOTO  NEXT         ; NO, clear next
CONTINUE                   ; YES, continue
```

有三种间接寻址寄存器。为了对整个数据存储器空间（4096 字节）寻址，这些寄存器的宽度都是 12 位。要存储 12 位的寻址信息，需要使用两个 8 位寄存器。这些间接寻址寄存器是：

1. FSR0：由 FSR0H:FSR0L 组成
2. FSR1：由 FSR1H:FSR1L 组成
3. FSR2：由 FSR2H:FSR2L 组成

此外还有寄存器 INDF0、INDF1 和 INDF2，它们都不是物理实现的。读写这些寄存器都会引起间接寻址，相应的 FSR 寄存器中的值是数据地址。如果指令向 INDF0 写入值，该值会被写入到 FSR0H:FSR0L 指向的地址。从 INDF1 读取数据，实际读取的是 FSR1H:FSR1L 指向的地址中的数据。只要代码中可使用操作数，就可以使用 INDFn。

如果通过 FSR 间接读取 INDF0、INDF1 或 INDF2，所有的读取将为 0（Z 位被置 1）。与此类似，如果间接写 INDF0、 INDF1 或 INDF2，操作等同于 `NOP` 指令，STATUS 位不受影响。

4.12.1 间接寻址操作

除了 4 个寄存器地址外，每个 FSR 寄存器都有与其相关的 INDF 寄存器。对这 5 个寄存器之一进行操作将决定间接寻址时修改 FSR 的方式。

当对这 5 个 INDFn 单元中的任何一个进行数据存取时，选择的地址将 FSRn 寄存器配置为：

- 在间接访问后对 FSRn 不做任何操作 （无变化）—INDFn
- 在间接访问后自动递减 FSRn 的值 （后减）—POSTDECn
- 在间接访问后自动递增 FSRn 的值 （后加）—POSTINCn
- 在间接访问前自动递增 FSRn 的值 （前加）－PREINCn
- 用 WREG 中的值作为 FSRn 的偏移量。在间接访问后不修改 WREG 或 FSRn 寄存器的值 （无变化）－ PLUSWn

当使用自动增减功能时，STATUS 寄存器不会反映出对 FSR 的影响。例如，如果间接寻址导致 FSR 等于 0，则 Z 位将不会置 1。

FSR 的递增或递减会影响全部 12 位。也就是说，如果 FSRnL 因为递增而上溢，则 FSRnH 将会自动递增。

增加这些功能可以让 FSRn 除了用于数据存储器中的表操作外，还可用作堆栈指针。

每个FSR 都有一个与之相关的地址，该地址用于进行索引间接存取。当对此 INDFn 单元 （PLUSWn）进行数据存取时，FSRn 将在进行间接存取前把 WREG 寄存器中的带符号值和 FSR 中的值相加以构成地址。 FSR 值不会改变。

如果某个 FSR 寄存器包含的值指向其中一个 INDFn，间接读取的值为 00h （Z 位置 1），而间接写则等同于 `NOP` （STATUS 位不受影响）。

如果执行间接寻址的目标地址为 FSRnH 或FSRnL 寄存器，写操作将影响前置递增 / 前置递减 / 后置递增 / 后置递减功能。

图 **4-9**： 间接寻址操作

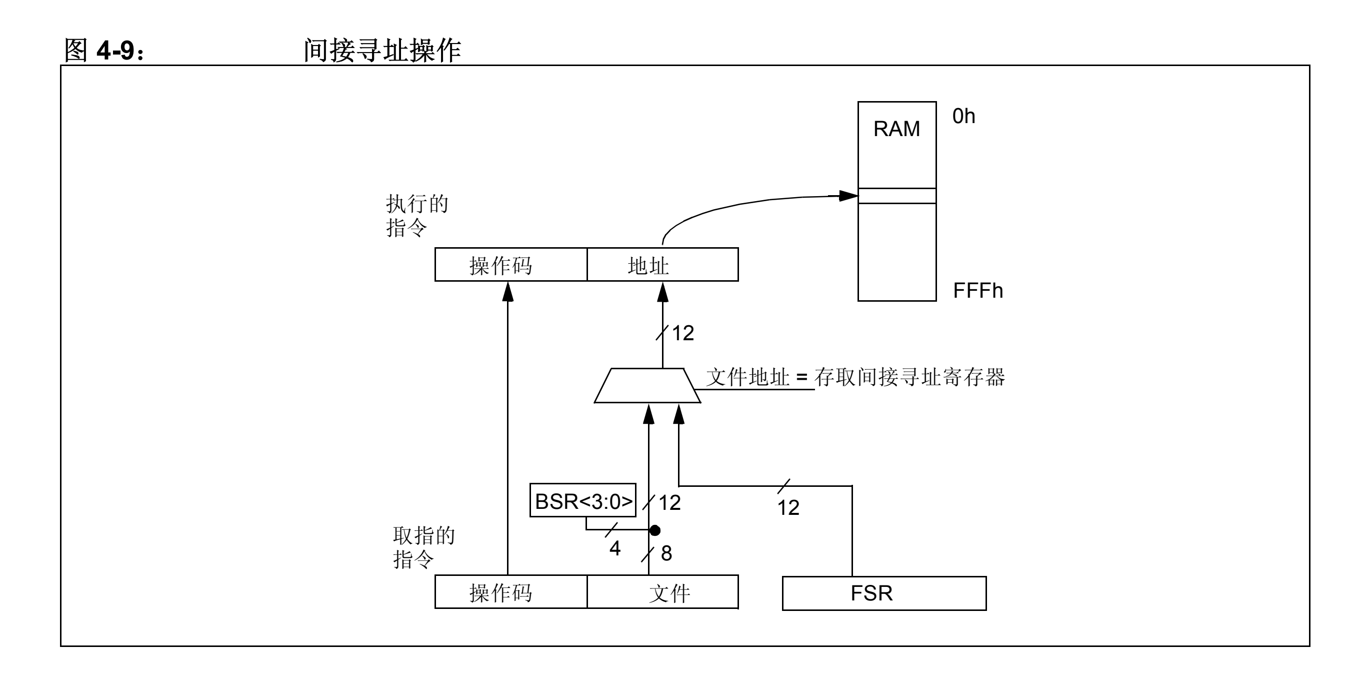
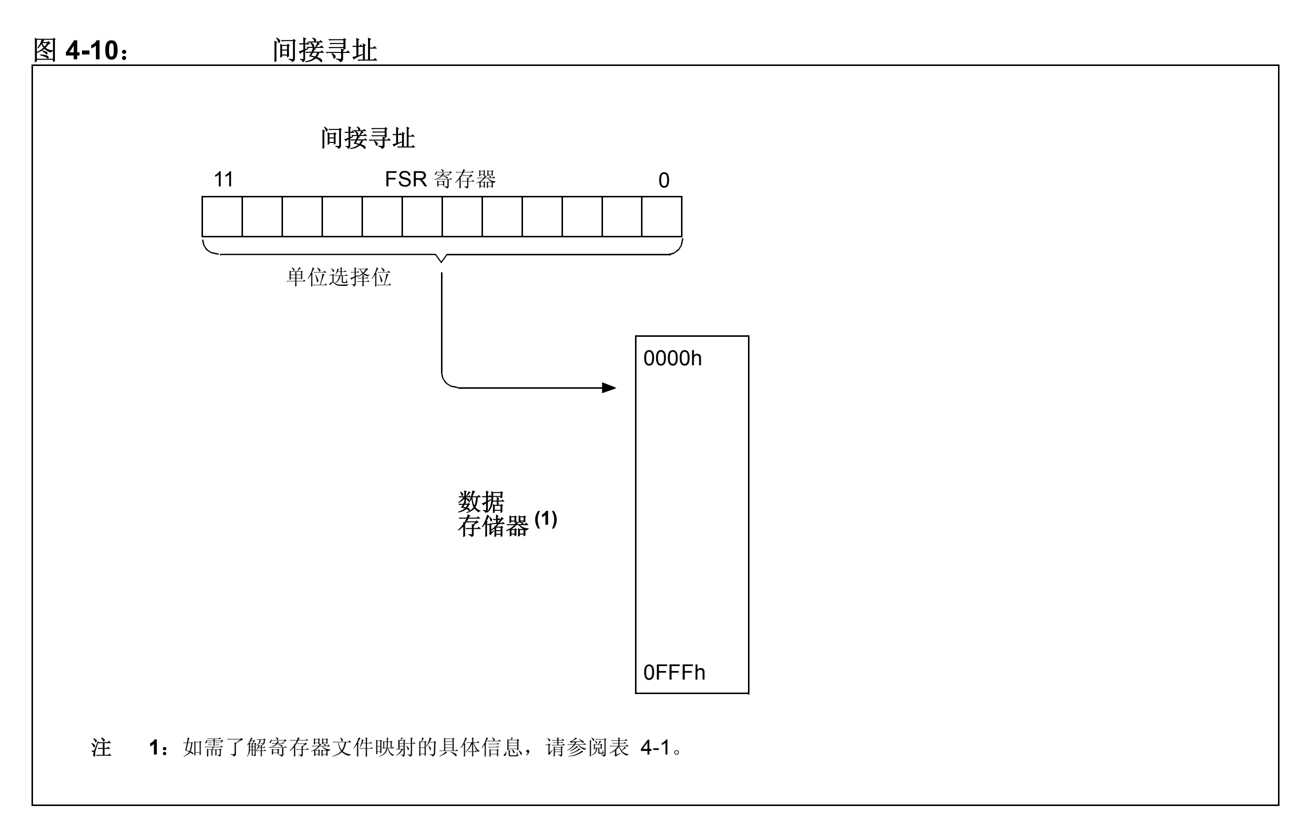


图 **4-10**： 间接寻址



注 **1**：如需了解寄存器文件映射的具体信息，请参阅表 4-1。

4.13 状态寄存器 （STATUS）

如寄存器 4-2所示，状态寄存器包含ALU的算术运算状态。状态寄存器和其他寄存器一样，可以作为任何指令的目标寄存器。如果一条影响 Z、DC、C、OV 或 N 位的指令的目标寄存器是状态寄存器，则会禁止对这 5 位进行直接写操作。根据器件逻辑，对这些位置1或清零。所以，当所执行指令的目标寄存器是状态寄存器时，状态寄存器的结果可能和预想的不一样。

例如，指令 `CLRF STATUS` 将会把状态寄存器的高 3 位清零，并将 Z 位置 1。操作后状态寄存器的结果为 `000u u1uu` （其中 u 表示未变化）。

因此，建议仅使用 BCF、BSF、SWAPF、MOVFF 和 MOVWF 指令来改变状态寄存器，因为这些指令不影响状态寄存器中的 Z、 C、 DC、 OV 或 N 位。关于其他不影响任何状态位的指令，请见表 20-2。

注：	在减法运算中，C 和 DC 位分别作为 $\overline{借位}$ 和辅助借位。

寄存器 4-2： 状态寄存器

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC	C
bit 7							bit 0

bit 7-5 **未实现：**读为 0

bit 4 **N：**负数标志位
此位用于有符号的算术运算 （2 进制补码）。表明结果是否为负数 （ALU MSB ＝ 1）。
1= 结果为负
0= 结果为正

bit 3 **OV：**溢出标志位
此位用于有符号的算术运算（2 进制补码）。表明 7 位数量级的溢出，溢出将导致符号位（bit7）改变状态。
1= 有符号算术运算发生溢出 （本次运算）
0= 没有发生溢出

bit 2 **Z：**零标志位
1= 算术或逻辑运算结果为零
0 = 算术或逻辑运算的结果非零

bit 1 **DC：**辅助进位 / $\overline{借位}$
用于 ADDWF、 ADDLW、 SUBLW 和 SUBWF 指令
1= 结果的第 4 低位发生了进位
0= 结果中第 4 低位没有发生进位

注： $\overline{辅助借位}$的极性是相反的。减法运算通过加上第二个操作数的 2 进制补码来实现。对于移位指令 （RRF 和 RLF），此位值来自源寄存器的 bit 4 或 bit 3。

bit 0 **C:** 进位 / $\overline{借位}$标志位
用于 ADDWF、 ADDLW、 SUBLW 和 SUBWF 指令
1= 结果中最高位发生了进位
0= 结果中最高位没有发生进位

注： $\overline{借位}$的极性是相反的。减法运算通过加上第二个操作数的 2 进制补码来实现。对于移位指令 （RRF 和 RLF），此位值来自源寄存器的最高位或最低位。

图注：			
R= 可读位	W= 可写位	U= 未实现位，读为 0	
-n= 上电复位时的值	1= 置位	0= 清零	x= 未知位

4.14 RCON 寄存器

复位控制（Reset Control， RCON）寄存器包含标志位，可以区分器件复位的来源。这些标志有 TO、 PD、POR、 BOR 和 RI 位。该寄存器是可读写的。

> 注 1：如果 BOREN 配置位置 1（使能欠压复位）， BOR 位在上电复位时为 1。发生了欠压复位后，BOR 位将被清零并必须由固件置 1 以指示下一次欠压复位。
>
> 2：建议在检测到上电复位时，将 $\overline{\text{POR}}$ 位置 1，以便检测后续的上电复位。

寄存器 4-3: RCON 寄存器

R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
bit 7							bit 0

bit 7 **IPEN**：中断优先级使能位
1= 使能中断优先级
0= 禁止中断优先级 （16CXXX 兼容模式）

bit 6-5 **未实现**：读为 0

bit 4 **$\overline{\text{RI}}$**：RESET 指令标志位
1= 未执行 RESET 指令
0= 执行了 RESET 指令，产生一个器件复位
（欠压复位后必须软件置 1）

bit 3 **$\overline{\text{TO}}$**：看门狗定时器超时溢出标志位
1= 上电后执行了 CLRWDT 指令或 SLEEP 指令
0= 发生了 WDT 超时

bit 2 **$\overline{\text{PD}}$**：掉电检测标志位
1= 上电或执行了 CLRWDT 指令后
0= 执行了 SLEEP

bit 1 **$\overline{\text{POR}}$**：上电复位状态位
1= 未发生上电复位
0= 发生了上电复位
（在上电复位后必须软件置 1）

bit 0 **$\overline{\text{BOR}}$**：欠压复位状态位
1= 未发生欠压复位
0= 发生了欠压复位
（欠压复位后必须软件置 1）

图注：			
R= 可读位	W= 可写位	U= 未实现位，读为 “0”	
-n= 上电复位时的值	1= 置位	0= 清零	x= 未知位

注：

5.0 闪存程序存储器

在整个 VDD 范围内的正常运行期间，闪存程序存储器可读写并且可擦除。

对程序存储器的读操作每次读取一个字节。对程序存储器的写操作每次写入 8 字节的块。每次以 64 字节的块擦除程序存储器。可能不允许用户代码执行批量擦除操作。

写或擦除程序存储器将停止指令获取，直到操作完成为止。在写或擦除期间不能访问程序存储器，因此无法执行代码。内部编程定时器可以结束程序存储器的写操作和擦除操作。

写入程序存储器的值不必是有效指令。执行一个程序存储器单元（该单元包含一个无效指令）将产生一个 NOP 指令。

5.1 读表和写表操作

为了能够读写程序存储器，有两种操作允许处理器在程序存储器空间和数据 RAM 之间移动字节。

- 读表操作 （TBLRD）
- 写表操作 （TBLWT）

程序存储器空间是 16 位宽，而数据 RAM 空间是 8 位宽。读表操作和写表操作通过一个 8 位寄存器（TABLAT）在这两个存储器空间之间移动数据。

读表操作从程序存储器中取出数据存入数据RAM空间。图 5-1 显示了在程序存储器和数据 RAM 之间执行的读表操作。

写表操作将数据存储器空间中的数据保存到程序存储器的保持寄存器中。第 5.5 节“写入闪存程序存储器”详细介绍了怎样将保持寄存器中的内容写入程序存储器。图 5-2 显示了在程序存储器和数据 RAM 间执行的写表操作。

表操作以字节为单位执行。一个包含数据而非程序指令的表块不必以字为单位。因此，表块可以在任何字节地址开始和结束。如果使用写表操作将可执行代码写入程序存储器，程序指令就需要以字为单位。

图 5-1： 读表操作

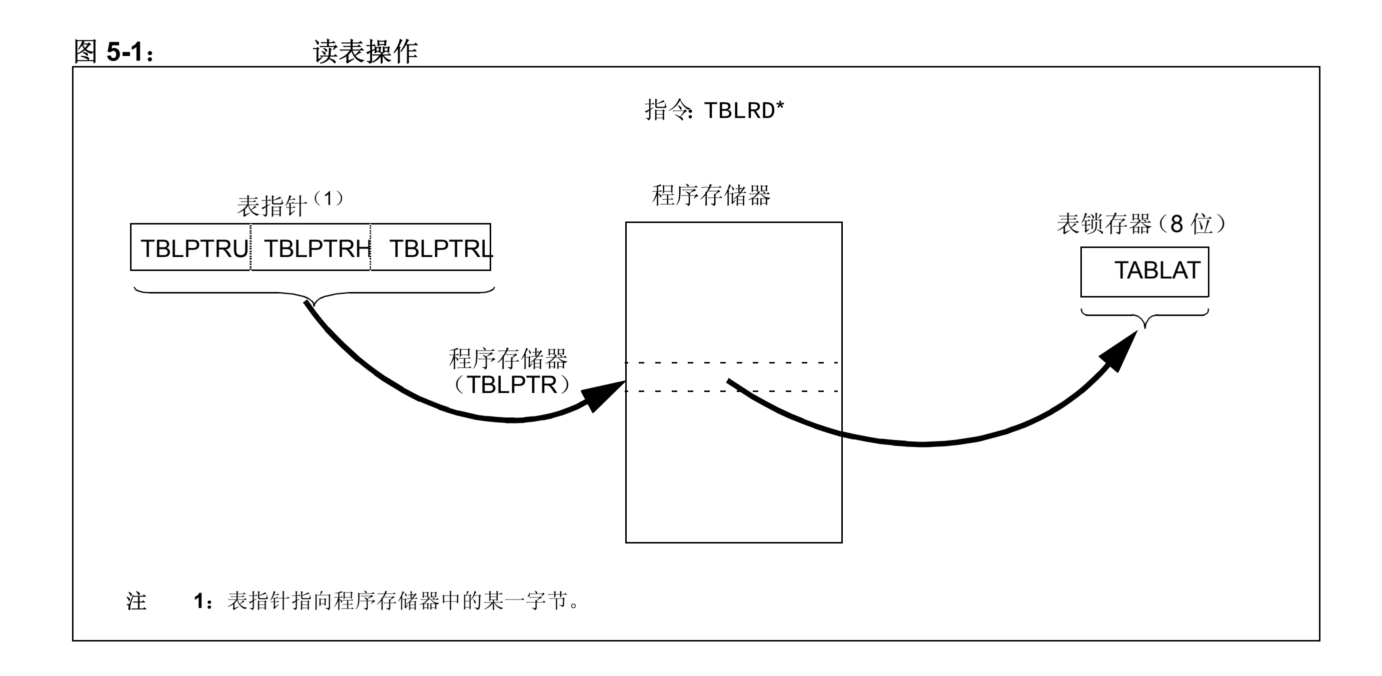
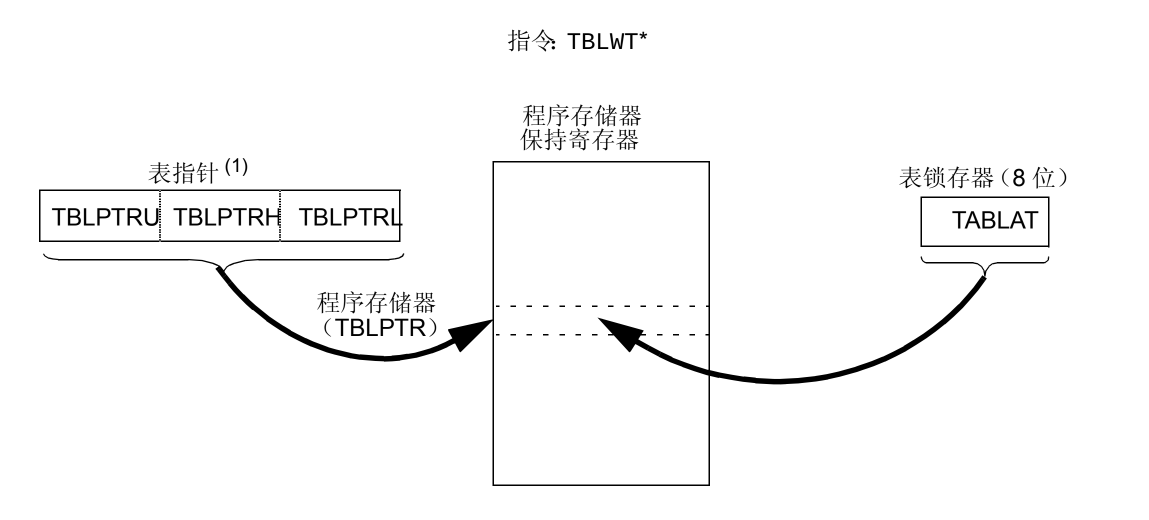


图 5-2： 写表操作



注 1：表指针实际指向 8 个保持寄存器中的一个，它的地址由 TBLPTRL<2:0> 决定。将数据物理写入程序存储器阵列的过程将在第 5.5 节中讨论。

5.2 控制寄存器

使用 TBLRD 与 TBLWT 指令时，会用到一些控制寄存器。它们包括：

- EECON1 寄存器
- EECON2 寄存器
- TABLAT 寄存器
- TBLPTR 寄存器

5.2.1 EECON1 和 EECON2 寄存器

EECON1 是针对存储器访问的控制寄存器。

EECON2 不是一个物理存在的寄存器。对 EECON2 读的结果是全"0"。EECON2 寄存器专门用于存储器的写和擦除时序。

控制位 EEPGD 决定将访问程序存储器还是数据 EEPROM 存储器。若该位清零，任何后续操作将针对数据 EEPROM 存储器进行。若该位置 1，任何后续操作将针对程序存储器进行。

控制位 CFGS 决定将访问配置寄存器还是程序存储器 / 数据 EEPROM 存储器。若该位置 1，则无论 EEPGD 如何设置，后续操作都将针对配置寄存器进行（请参见第 19.0 节"CPU 的特殊功能"）。若该位清零，存储器选择访问取决于 EEPGD。

若将 FREE 位置 1，则允许程序存储器擦除操作。若 FREE 位为置位状态，在下一个 WR 命令时将执行擦除操作。若 FREE 清零，只能进行写操作。

若将 WREN 位置 1，则允许写操作。上电时将清零 WREN 位。正常运行期间，当写操作因 $\overline{\text{MCLR}}$ 复位或 WDT 超时复位而中断时，WRERR 位置 1。这些情况发生时，用户可检查 WRERR 位，并对该单元进行重写。由于 RESET 值为零，必须重新装入数据并寻址寄存器（EEDATA 和 EEADR）。

控制位 WR 用于启动写操作。此位不能被软件清零，只能用软件置 1。写操作完成后，由硬件进行清零。因为软件无法对 WR 位清零，所以不会意外地或过早地终止写操作。

> **注：** 写操作完成后，PIR2 寄存器里的中断标志位 EEIF 被置 1。它必须用软件清零。

寄存器 5-1： EECON1 寄存器 （地址 FA6h）

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/W-0	R/W-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7 **EEPGD**：闪存程序存储器或数据 EEPROM 存储器选择位
1= 访问闪存程序存储器
0= 访问数据 EEPROM 存储器

bit 6 **CFGS**：闪存程序 / 数据 EE 或配置选择位
1= 访问配置寄存器
0= 访问闪存程序存储器或数据 EEPROM 存储器

bit 5 **未实现**：读为 0

bit 4 **FREE**：闪存行擦除使能位
1= 在下一个 WR 命令时擦除由 TBLPTR 寻址的程序存储器行
（擦除操作完成时清零）
0= 只执行写操作

bit 3 **WRERR**：闪存程序 / 数据 EE 错误标志位
1 = 写操作提前终止
（正常运行时自定时编程设计中的任何复位）
0= 写操作完成

注： 发生 WRERR 时， EEPGD 和 CFGS 位不清零。这样可以跟踪错误状态。

bit 2 **WREN**：闪存程序 / 数据 EE 写使能位
1= 允许写周期
0= 禁止写入 EEPROM

bit 1 **WR**：写操作控制位
1= 启动数据 EEPROM 的擦写周期或者程序存储器的擦写周期。
（这是一个自定时操作，一旦完成写操作，该位将被硬件清零。 WR 位只能由软件置 1，而不能用软件清零。）
0= 对 EEPROM 的写周期已完成

bit 0 **RD**：读操作控制位
1= 启动 EEPROM 读操作
（读操作需要一个周期。 RD 由硬件清零。 RD 位只能用软件置 1，而不能用软件清零。当 EEPGD=1 时， RD 位不能被置 1。）
0= 未启动 EEPROM 读操作

图注：			
R= 可读位	W= 可写位	U= 未实现位，读为 0	
-n= 上电复位时的值	1= 复位	0= 清零	x= 位状态未知

5.2.2 TABLAT——表锁存寄存器

表锁存器（Table Latch，TABLAT）是映射到 SFR 空间的 8 位寄存器。表锁存器用来在程序存储器和数据 RAM 之间进行数据转移时保存 8 位数据。

5.2.3 TBLPTR——表指针寄存器

表指针 （Table Pointer， TBLPTR）在程序存储器中寻址字节。 TBLPTR 由三个 SFR 寄存器组成：表指针最高位字节、表指针高位字节和表指针低位字节（TBLPTRU:TBLPTRH:TBLPTRL） 。这三个寄存器一起组成了一个 22 位宽的指针。其中低 21 位允许器件可寻址至多 2MB 的程序存储器空间。第 22 位允许访问器件 ID、用户 ID 以及配置位。

表指针 TBLPTR 由 TBLRD 和 TBLWT 指令使用。利用四种表操作方法中的一种，这些指令可以更新 TBLPTR。表 5-1 列出了这些操作。对 TBLPTR 执行这些操作仅仅影响低 21 位。

5.2.4 表指针边界

TBLPTR 用于读写和擦除闪存程序存储器。

执行 TBLRD 时，表指针的所有 22 位决定将程序存储器中的哪个字节读入 TABLAT。

执行 TBLWT 时，表指针的三个 LSb （TBLPTR<2:0>）决定将写入程序存储器 8 个保持寄存器中的哪一个。当开始定时写入程序存储器（长写周期）时，表指针 TBLPTR 的 19 个 MSb（TBLPTR<21:3>）决定将写入程序存储器中的哪个 8 字节块。若需了解更多详情，请参见第 5.5 节 （“写入闪存程序存储器”）。

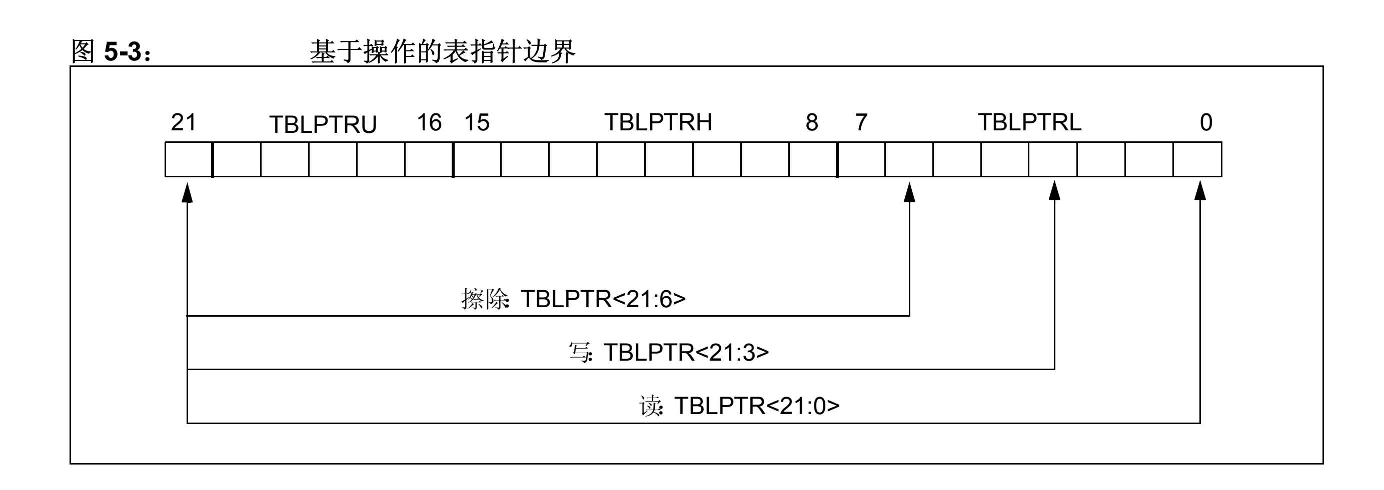
执行擦除程序存储器时，表指针的 16 个 MSb（TBLPTR<21:6>）指向要擦除的 64 字节块。忽略最低有效位 （TBLPTR<5:0>）。

图 5-3 描述了基于闪存程序存储器操作的相关 TBLPTR 边界。

表 5-1： 用 TBLRD 和 TBLWT 指令执行表指针操作

示例	表指针操作
TBLRD* TBLWT*	不修改 TBLPTR
TBLRD*+ TBLWT*+	读 / 写操作后递增 TBLPTR
TBLRD*- TBLWT*-	读 / 写操作后递减 TBLPTR
TBLRD+* TBLWT+*	读 / 写操作前递增 TBLPTR

图 5-3： 基于操作的表指针边界



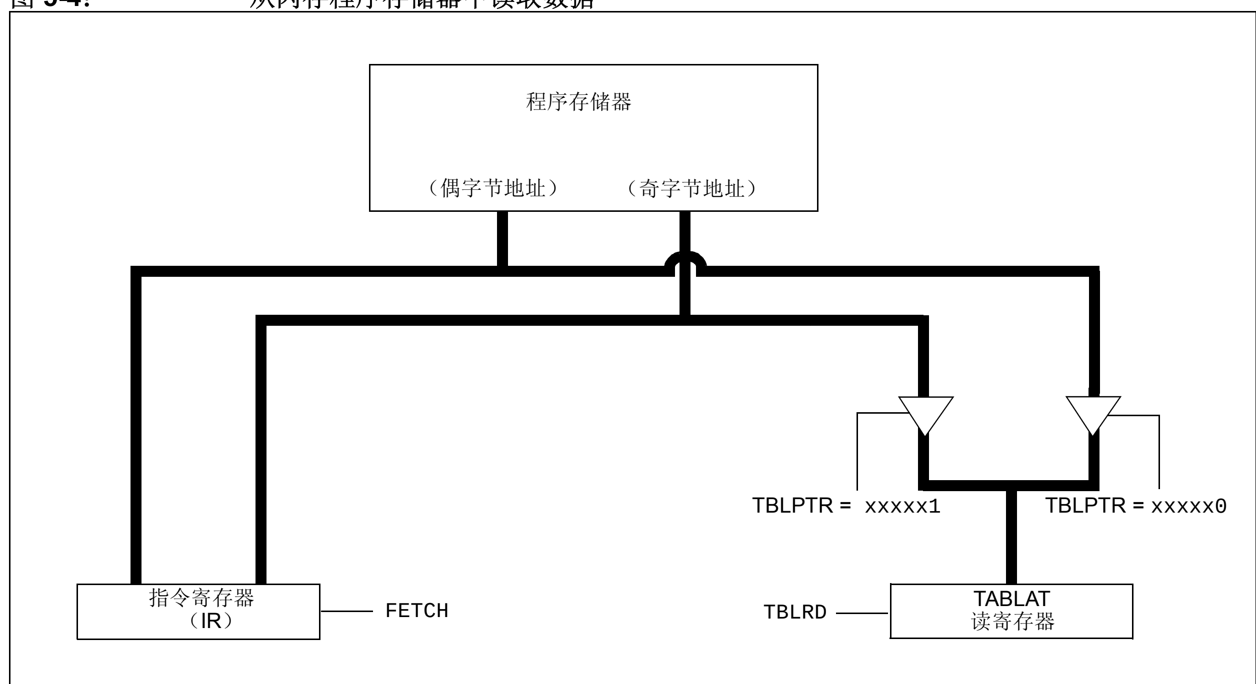
5.3 读取闪存程序存储器

TBLRD 指令用于从程序存储器中获得数据并存入数据 RAM。读表操作每次从程序存储器读取一个字节。

TBLPTR 指向程序空间中的一个字节地址。执行 TBLRD 将所指向的字节装入 TABLAT。另外，可以自动修改 TBLPTR 以进行下次读表操作。

内部程序存储器通常以字为单位操作。由地址的最低有效位来选择字的高字节还是低字节。图 5-4 显示了内部程序存储器和 TABLAT 之间的交互。

图 5-4： 从闪存程序存储器中读取数据



例 5-1： 读取一个闪存程序存储器字

```
            MOVLW CODE_ADDR_UPPER         ; Load TBLPTR with the base
            MOVWF TBLPTRU                 ; address of the word
            MOVLW CODE_ADDR_HIGH
            MOVWF TBLPTRH
            MOVLW CODE_ADDR_LOW
            MOVWF TBLPTRL
READ_WORD
            TBLRD*+                       ; read into TABLAT and increment
            MOVF TABLAT, W                ; get data
            MOVWF WORD_EVEN
            TBLRD*+                       ; read into TABLAT and increment
            MOVF TABLAT, W                ; get data
            MOVWF WORD_ODD
```


5.4 擦除闪存程序存储器

最小擦除块是 32 字，即 64 字节。只有通过使用外部编程器或者 ICSP 控制才能批量擦除数个较大的程序存储器块。闪存阵列中不支持字擦除。

当从单片机本身启动擦除时序时，将擦除程序存储器的一个 64 字节块。TBLPTR<21:6> 的高 16 位指向将被擦除的块。忽略 TBLPTR<5:0>。

EECON1 寄存器控制擦除操作。必须把 EEPGD 位置 1 以指向闪存程序存储器。必须把 WREN 位置 1 以使能写操作。把 FREE 位置 1 以选择擦除操作。

为了保护起见，必须使用 EECON2 的写操作启动时序。

要擦除内部闪存，必须执行长写操作。在长写周期中会暂停指令的执行。内部编程定时器将终止这个长写周期。

5.4.1 闪存程序存储器擦除操作顺序

擦除内部程序存储器单元块的事件顺序：

1. 把要擦除的行地址装载到表指针。
2. 将 EEPGD 位置 1，指向程序存储器，并将 CFGS 位清零以访问程序存储器，再将 WREN 位置 1 以使能写操作，然后将 FREE 位置 1 以启用擦除操作。
3. 禁用中断。
4. 将 55h 写入 EECON2。
5. 将 AAh 写入 EECON2。
6. 把 WR 位置 1。这将开始行擦除周期。
7. 在擦除操作期间， CPU 将停止工作 （内部定时器计时 2ms 左右）。
8. 重新使能中断。

例 5-2： 擦除闪存程序存储器行

```
                    MOVLW   CODE_ADDR_UPPER       ; load TBLPTR with the base
                    MOVWF   TBLPTRU               ; address of the memory block
                    MOVLW   CODE_ADDR_HIGH
                    MOVWF   TBLPTRH
                    MOVLW   CODE_ADDR_LOW
                    MOVWF   TBLPTRL
          ERASE_ROW
                    BSF     EECON1,EEPGD          ; point to FLASH program memory
                    BCF     EECON1,CFGS           ; access FLASH program memory
                    BSF     EECON1,WREN           ; enable write to memory
                    BSF     EECON1,FREE           ; enable Row Erase operation
                    BCF     INTCON,GIE            ; disable interrupts
                    MOVLW   55h
    必须按照        MOVWF   EECON2                ; write 55h
    这个顺序        MOVLW   AAh
                    MOVWF   EECON2                ; write AAh
                    BSF     EECON1,WR             ; start erase (CPU stall)
                    BSF     INTCON,GIE            ; re-enable interrupts
```


5.5 写入闪存程序存储器

最小编程块是 4 字，即 8 字节。不支持字或字节编程。

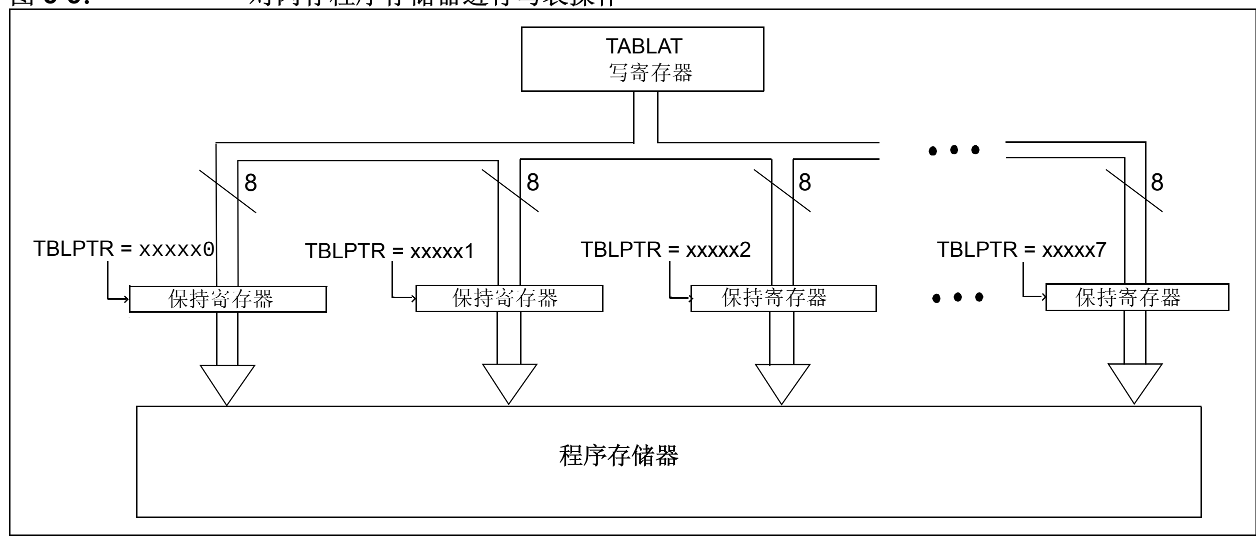
写表操作用于在内部装入编程闪存存储器所需的保持寄存器。写表操作使用 8 个保持寄存器编程。

因为表锁存器 （TABLAT）仅是单字节操作，所以每个编程操作必须执行 8 次 TBLWT 指令。由于只写入保持寄存器，所有写表操作实际上都是短写周期。在更新完 8 个寄存器后，必须向 EECON1 寄存器写入数据，以开始一个长写周期的编程操作。

要编程内部闪存，必须执行长写操作。在长写周期中将暂停指令的执行。内部编程定时器将终止这个长写周期。

EEPROM 片上定时器控制写操作时间。在字或字节操作中，写 / 擦除电压由在器件的工作电压范围内运行的片上电荷泵产生。

图 5-5： 对闪存程序存储器进行写表操作



5.5.1 闪存程序存储器写操作顺序

编程内部程序存储器单元的事件顺序应该是：

1. 读 64 字节数据到 RAM。
2. 必要时更新 RAM 中的数值。
3. 把将要擦除的地址装入表指针。
4. 执行行擦除。
5. 把要写入的第一个字节地址装入表指针。
6. 通过自动递增 （TBLWT*+ 或 TBLWT+*）将头 8 个字节写入保持寄存器。
7. 将EEPGD位置1以指向程序存储器，清零CFGS 位以访问程序寄存器，同时将 WREN 置 1，使能字节写操作。
8. 禁用中断。
9. 将 55h 写入 EECON2。
10. 将 AAh 写入 EECON2。
11. 将 WR 位置 1。这将开始写周期。
12. 在写操作期间， CPU 将停止工作 （内部定时器计时 2ms 左右）。
13. 重新使能中断。
14. 重复步骤 6-14 共 7 次，写入 64 字节。
15. 校验存储器 （读表操作）。

此过程大约需要18ms来更新存储器的一行（64字节）。例 5-3 给出了必要的代码示例。

注：	在把 WR 位置 1 之前，表指针地址应在保持寄存器中期望的 8 字节地址范围内。

例 **5-3：** 向闪存程序存储器写入数据

```
            MOVLW   D'64                        ; number of bytes in erase block
            MOVWF   COUNTER
            MOVLW   BUFFER_ADDR_HIGH            ; point to buffer
            MOVWF   FSR0H
            MOVLW   BUFFER_ADDR_LOW
            MOVWF   FSR0L
            MOVLW   CODE_ADDR_UPPER             ; Load TBLPTR with the base
            MOVWF   TBLPTRU                     ; address of the memory block
            MOVLW   CODE_ADDR_HIGH
            MOVWF   TBLPTRH
            MOVLW   CODE_ADDR_LOW
            MOVWF   TBLPTRL
READ_BLOCK
            TBLRD*+                             ; read into TABLAT, and inc
            MOVF    TABLAT, W                   ; get data
            MOVWF   POSTINC0                    ; store data
            DECFSZ  COUNTER                     ; done?
            BRA     READ_BLOCK                  ; repeat
MODIFY_WORD
            MOVLW   DATA_ADDR_HIGH              ; point to buffer
            MOVWF   FSR0H
            MOVLW   DATA_ADDR_LOW
            MOVWF   FSR0L
            MOVLW   NEW_DATA_LOW                ; update buffer word
            MOVWF   POSTINC0
            MOVLW   NEW_DATA_HIGH
            MOVWF   INDF0
ERASE_BLOCK
            MOVLW   CODE_ADDR_UPPER             ; load TBLPTR with the base
            MOVWF   TBLPTRU                     ; address of the memory block
            MOVLW   CODE_ADDR_HIGH
            MOVWF   TBLPTRH
            MOVLW   CODE_ADDR_LOW
            MOVWF   TBLPTRL
            BSF     EECON1,EEPGD                ; point to FLASH program memory
            BCF     EECON1,CFGS                 ; access FLASH program memory
            BSF     EECON1,WREN                 ; enable write to memory
            BSF     EECON1,FREE                 ; enable Row Erase operation
            BCF     INTCON,GIE                  ; disable interrupts
            MOVLW   55h
            MOVWF   EECON2                      ; write 55h
            MOVLW   AAh
            MOVWF   EECON2                      ; write AAh
            BSF     EECON1,WR                   ; start erase (CPU stall)
            BSF     INTCON,GIE                  ; re-enable interrupts
            TBLRD*-                             ; dummy read decrement
WRITE_BUFFER_BACK
            MOVLW   8                           ; number of write buffer groups of 8 bytes
            MOVWF   COUNTER_HI
            MOVLW   BUFFER_ADDR_HIGH            ; point to buffer
            MOVWF   FSR0H
            MOVLW   BUFFER_ADDR_LOW
            MOVWF   FSR0L
PROGRAM_LOOP
            MOVLW   8                           ; number of bytes in holding register
            MOVWF   COUNTER
WRITE_WORD_TO_HREGS
            MOVF    POSTINC0, W                 ; get low byte of buffer data
            MOVWF   TABLAT                      ; present data to table latch
            TBLWT+*                             ; write data, perform a short write
                                                ; to internal TBLWT holding register.
            DECFSZ  COUNTER                     ; loop until buffers are full
            BRA     WRITE_WORD_TO_HREGS
```


例 5-3： 向闪存程序存储器写入数据 （续）

```
PROGRAM_MEMORY
            BSF     EECON1,EEPGD          ; point to FLASH program memory
            BCF     EECON1,CFGS           ; access FLASH program memory
            BSF     EECON1,WREN           ; enable write to memory
            BCF     INTCON,GIE            ; disable interrupts
            MOVLW   55h
必须按照    MOVWF   EECON2                ; write 55h
这个顺序    MOVLW   AAh
            MOVWF   EECON2                ; write AAh
            BSF     EECON1,WR             ; start program (CPU stall)
            BSF     INTCON,GIE            ; re-enable interrupts
            DECFSZ  COUNTER_HI            ; loop until done
            BRA     PROGRAM_LOOP
            BCF     EECON1,WREN           ; disable write to memory
```

5.5.2 写校验

根据应用的需要，好的编程习惯一般需要对照原始值验证已写入存储器的值。应用中，当过度的写操作接近参数极限值时，就应进行写校验。

5.5.3 写操作异常终止

如果意外事件 （例如掉电或者异常复位）终止了写操作，则应该校验刚刚编程的存储器单元，必要时需要重新编程。正常运行期间，如果 MCLR 复位或 WDT 超时复位而中断了写操作，则 WRERR 位置 1。在这些情况下，用户可以检查 WRERR 位，并重写此单元。

5.5.4 防止误写

为了防止对闪存程序存储器进行误写操作，必须遵循写操作启动顺序。如需了解更多详情，请参见 "CPU 的特殊功能"（第 19.0 节）。

5.6 闪存程序操作代码保护功能

如需了解更多关于闪存程序存储器代码保护的详情，请参阅第 19.0 节 "CPU 的特殊功能"。

表 5-2： 与程序闪存存储器有关的寄存器

地址	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 或 BOR 时的值	其他所有复位时的值
FF8h	TBLPTRU	—	—	bit21	程序存储器表指针最高位字节 （TBLPTR<20:16>）					--00 0000	--00 0000
FF7h	TBPLTRH	程序存储器表指针高位字节 （TBLPTR<15:8>）								0000 0000	0000 0000
FF6h	TBLPTRL	程序存储器表指针低位字节 （TBLPTR<7:0>）								0000 0000	0000 0000
FF5h	TABLAT	程序存储器表锁存器								0000 0000	0000 0000
FF2h	INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	0000 000x	0000 000u
FA7h	EECON2	EEPROM 控制寄存器 2 （不是物理存在的寄存器）								—	—
FA6h	EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	uu-0 u000
FA2h	IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	---1 1111
FA1h	PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	---0 0000
FA0h	PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	---0 0000

图注： x= 未知， u= 不变， r= 保留， -= 未实现，读为 '0'。
阴影单元在访问闪存 /EEPROM 时没有用到。

注：

6.0 数据 EEPROM 存储器

在整个 V_{DD} 范围内的正常运行期间，数据 EEPROM 存储器是可读写的。该数据存储器并不直接映射到寄存器文件空间，而是通过特殊功能寄存器（Special Function Registers， SFR）来间接寻址。

用四个 SFR 控制程序和数据 EEPROM 存储器的读写，它们是：

- EECON1
- EECON2
- EEDATA
- EEADR

EEPROM 数据存储器可按字节进行读写。在与数据存储器块交互时，EEDATA 寄存器存放 8 位读写数据，而 EEADR 寄存器存放要访问的 EEPROM 单元地址。这些器件具有 256 个字节的数据 EEPROM，地址范围从 0h 到 FFh。

EEPROM 是具有高擦写周期的数据存储器。写入一个字节的操作将自动擦除该单元并写入新值（先擦除后写入）。写操作的时间由片内定时器控制。它随着电压、温度以及芯片本身的变化而变化。请参阅参数 D122（第 22.0 节 “电气特性”）以获得详细信息。

6.1 EEADR

该地址寄存器最多可寻址256个字节的数据EEPROM。

6.2 EECON1 寄存器和 EECON2 寄存器

EECON1 是针对 EEPROM 存储器访问的控制寄存器。

EECON2 不是一个物理存在的寄存器。对 EECON2 读的结果是全“0”。EECON2 寄存器专门用于 EEPROM 写操作时序。

控制位 RD 和 WR 分别用于启动读操作和写操作。软件不能将这些位清零，只能置 1。在读或写操作完成后，它们被硬件清零。由于软件无法对 WR 位清零，写操作不会意外地过早终止。

WREN 位置 1 后将允许一次写操作。上电时， WREN 位被清零。正常运行时如果写操作因 $\overline{MCLR}$ 复位或WDT 超时复位而中断，则WRERR位置1。这些情况发生时，用户可以检查 WRERR 位并重写该单元。还需要重新装入数据寄存器和地址寄存器 （EEDATA 和 EEADR），因为复位状态强制将这些寄存器的内容全部清零。

注：	当写操作完成时， PIR2 寄存器中的中断标志位 EEIF 会置 1。该位必须用软件清零。

寄存器 6-1：　EECON1 寄存器 （地址 FA6h）

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7　**EEPGD**：闪存程序或数据 EEPROM 存储器选择位
1= 访问闪存程序存储器
0= 访问数据 EEPROM 存储器

bit 6　**CFGS**：闪存程序 / 数据 EE 或配置选择位
1= 访问配置或校准寄存器
0= 访问闪存程序或数据 EEPROM 存储器

bit 5　**未实现**：读为 0

bit 4　**FREE**：闪存行擦除使能位
1= 在下一个 WR 命令时擦除由 TBLPTR 寻址的程序存储器行
（擦除操作完成时清零）
0= 仅执行写操作

bit 3　**WRERR**：闪存程序 / 数据 EE 错误标志位
1= 写操作过早终止
（正常运行时自定时编程设计中任何 $\overline{\text{MCLR}}$ 或 WDT 复位）
0= 写操作完成

注：　发生 WRERR 时， EEPGD 或 FREE 位不清零。这样可以跟踪错误状态。

bit 2　**WREN**：闪存程序 / 数据 EE 写使能位
1= 允许写周期
0= 禁止写入 EEPROM

bit 1　**WR**：写控制位
1= 启动数据 EEPROM 的擦写周期或者程序存储器的擦写周期。
（这是一个自定时操作，一旦完成写操作，该位将被硬件清零。 WR 位只能用软件置 1，而不能用软件清零。）
0 = 对 EEPROM 的写周期已完成

bit 0　**RD**：读控制位
1= 启动 EEPROM 读周期
（读操作需要一个周期。 RD 由硬件清零。 RD 位只能用软件置 1，而不能用软件清零。当 EEPGD=1 时， RD 位不能被置 1。）
0= 不启动 EEPROM 读操作

图注：			
R = 可读位	W= 可写位	U = 未实现位，读作 0	
-n = POR 时的值	1= 置位	0= 清零	x= 未知位

6.3 读取数据 EEPROM 存储器

要读取数据存储器单元，用户必须先把地址写入 EEADR 寄存器，对 EEPGD 控制位（EECON1<7>）清零，对 CFGS 控制位（EECON1<6>）清零，并将控制位 RD（EECON1<0>）置 1。接下来的一个指令周期就可以使用该数据；因此，EEDATA 寄存器可由下一条指令读取。 EEDATA 将该值一直保留到下一次读操作，或直到用户对该寄存器写入数据（在写操作期间）为止。

例 6-1： 数据 EEPROM 读操作

```
MOVLW   DATA_EE_ADDR   ;
MOVWF   EEADR          ; Data Memory Address to read
BCF     EECON1, EEPGD  ; Point to DATA memory
BCF     EECON1, CFGS   ; Access program FLASH or Data EEPROM memory
BSF     EECON1, RD     ; EEPROM Read
MOVF    EEDATA, W      ; W = EEDATA
```

6.4 向 EEPROM 数据存储器中写入

要对一个 EEPROM 数据单元进行写操作，必须首先将地址写入 EEADR 寄存器，再将数据写入 EEDATA 寄存器。然后必须按照例 6-2 中的次序来启动写周期。

如果没有完全按照例 6-2 中的次序执行（将 55h 写入 EECON2，将 AAh 写入 EECON2，然后将 WR 位置 1），写操作将不会开始。强烈建议在这一代码段中禁止中断。

此外， EECON1 的 WREN 位必须置 1 来使能写操作。这种机制可防止由于执行了意外的代码（即，程序跑飞）而错误地对数据 EEPROM 执行了写操作。除非在更新EEPROM，否则WREN位应始终保持为0；WREN 位不能由硬件清零。

写顺序开始以后，不能修改 EECON1、 EEADR 和 EDATA。除非将 WREN 置 1，否则不能将 WR 位置 1。必须在前一条指令中将 WREN 位置 1。 WR 和 WREN 不能在同一条指令中置 1。

写周期完成后， WR 位将被硬件清零，同时 EEPROM 写入完成中断标志位（EEIF）被置 1。用户可以使能此中断或对该位进行轮询。 EEIF 必须用软件清零。

例 6-2： 数据 EEPROM 写操作

```
            MOVLW   DATA_EE_ADDR   ;
            MOVWF   EEADR          ; Data Memory Address to read
            MOVLW   DATA_EE_DATA   ;
            MOVWF   EEDATA         ; Data Memory Value to write
            BCF     EECON1, EEPGD  ; Point to DATA memory
            BCF     EECON1, CFGS   ; Access program FLASH or Data EEPROM memory
            BSF     EECON1, WREN   ; Enable writes


            BCF     INTCON, GIE    ; Disable interrupts
必须按照    MOVLW   55h            ;
这个顺序    MOVWF   EECON2         ; Write 55h
            MOVLW   AAh            ;
            MOVWF   EECON2         ; Write AAh
            BSF     EECON1, WR     ; Set WR bit to begin write
            BSF     INTCON, GIE    ; Enable interrupts


            .                      ; user code execution
            .
            .
            BCF     EECON1, WREN   ; Disable writes on write complete (EEIF set)
```

6.5 写校验

根据具体应用，一个良好的编程习惯可能要求对照原始值来验证写入存储器的值。应用中，当过度的写操作接近参数极限值时，就应进行写校验。

6.6 防止误写

有些情况下，器件可能不希望向数据 EEPROM 存储器写入数据。为了防止 EEPROM 误写操作，器件内建了各种保护机制。上电时，WREN 位被清零。同时，上电定时器（持续时间 72 ms）也可防止误写 EEPROM。

在欠压、电源毛刺或软件故障期间，写操作启动顺序和 WREN 位可共同避免发生误写操作。

6.7 代码保护操作

数据 EEPROM 存储器具有自己的代码保护机制。只要启用其中一种机制，就会禁止外部读写操作。

单片机本身可以读写内部数据 EEPROM，而与代码保护配置位的状态无关。请参见第 19.0 节 “CPU 的特殊功能”以获取更多信息。

6.8 使用数据 EEPROM

数据 EEPROM 是一个高可靠性，并通过字节寻址的阵列，它被优化用来保存经常变更的信息（例如，程序变量或其他经常更新的数据）。经常更改的值的更新频率通常要高于 D124 规范中的规定。如果情况并非如此，必须执行阵列刷新。因此，不常更改的变量（例如常量、ID 及校准等）应该存储在闪存程序存储器中。

一个简单的数据 EEPROM 刷新程序如例 6-3 所示。

> **注：** 如果数据 EEPROM 仅用于保存常量和 / 或很少改变的数据，可能就不需要阵列刷新。请参见 D124 规范。

例 6-3： 数据 EEPROM 更新程序

```
        clrf    EEADR               ; Start at address 0
        bcf     EECON1,CFGS         ; Set for memory
        bcf     EECON1,EEPGD        ; Set for Data EEPROM
        bcf     INTCON,GIE          ; Disable interrupts
        bsf     EECON1,WREN         ; Enable writes
Loop                                ; Loop to refresh array
        bsf     EECON1,RD           ; Read current address
        movlw   55h                 ;
        movwf   EECON2              ; Write 55h
        movlw   AAh                 ;
        movwf   EECON2              ; Write AAh
        bsf     EECON1,WR           ; Set WR bit to begin write
        btfsc   EECON1,WR           ; Wait for write to complete
        bra     $-2
        incfsz  EEADR,F             ; Increment address
        bra     Loop                ; Not zero, do it again

        bcf     EECON1,WREN         ; Disable writes
        bsf     INTCON,GIE          ; Enable interrupts
```

表 **6-1**： 与数据 **EEPROM** 存储器有关的寄存器

地址	名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 或 BOR 时的值	其他所有复位时的值
FF2h	INTCON	GIE/GIEH	PEIE/GIEL	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
FA9h	EEADR	EEPROM 地址寄存器								0000 0000	0000 0000
FA8h	EEDATA	EEPROM 数据寄存器								0000 0000	0000 0000
FA7h	EECON2	EEPROM 控制寄存器 2 （非物理存在的寄存器）								—	—
FA6h	EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	uu-0 u000
FA2h	IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	---1 1111
FA1h	PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	---0 0000
FA0h	PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	---0 0000

图注： x= 未知， u =不变， r= 保留， -= 未实现，读为 0
阴影单元在访问闪存 /EEPROM 时没有用到。

注：

7.0 8 X 8 硬件乘法器

7.1 简介

8 x 8 硬件乘法器包含在 PIC18FXX2 器件的 ALU 中。它通过对硬件的操作，在一个单指令周期内实现乘法运算。得到 16 位的无符号运算结果，并存储到 16 位乘积寄存器对（PRODH:PRODL）中。乘法器不会影响 ALUSTA 寄存器中任何标志位。

在一个周期内使用 8 x 8 乘法器执行运算具有以下优点：

- 较高的计算吞吐量
- 减少乘法运算所需的代码量

性能的增强使得这个器件可以用于实现以前只能用数字信号处理器实现的应用。

表 7-1 显示了采用单周期硬件乘法器的增强型器件，和无硬件乘法器的情况下，实现同样功能时两者性能的比较。

表 7-1: 性能比较

程序	乘法方式	程序存储器（字）	周期数（最大值）	时间		
				@ 40 MHz	@ 10 MHz	@ 4 MHz
8 x 8 无符号	不支持硬件乘法	13	69	6.9 ms	27.6 ms	69 ms
	硬件乘法	1	1	100 ns	400 ns	1 ms
8 x 8 有符号	不支持硬件乘法	33	91	9.1 ms	36.4 ms	91 ms
	硬件乘法	6	6	600 ns	2.4 ms	6 ms
16 x 16 无符号	不支持硬件乘法	21	242	24.2 ms	96.8 ms	242 ms
	硬件乘法	24	24	2.4 ms	9.6 ms	24 ms
16 x 16 有符号	不支持硬件乘法	52	254	25.4 ms	102.6 ms	254 ms
	硬件乘法	36	36	3.6 ms	14.4 ms	36 ms

7.2 操作

例 7-1 给出实现一个 8 x 8 无符号乘法的指令序列。当其中一个自变量已经装载到 WREG 寄存器时，只需一个指令周期就可以实现乘法运算。

例 7-2 给出了实现 8 x 8 有符号乘法的指令序列。要考虑自变量的符号位，就要测试每个自变量的最高有效位（Most Significant bit， MSb）并完成正确的减法运算。

例 7-1： 8 x 8 无符号乘法程序

```
MOVF     ARG1, W     ;
MULWF    ARG2        ; ARG1 * ARG2 ->
                     ;   PRODH:PRODL
```

例 7-2： 8 x 8 有符号乘法程序

```
MOVF     ARG1,  W
MULWF    ARG2          ; ARG1 * ARG2 ->
                       ;   PRODH:PRODL
BTFSC    ARG2,  SB     ; Test Sign Bit
SUBWF    PRODH, F      ; PRODH = PRODH
                       ;         - ARG1
MOVF     ARG1,  W
BTFSC    ARG2,  SB     ; Test Sign Bit
SUBWF    PRODH, F      ; PRODH = PRODH
                       ;         - ARG1
```

例 7-3 给出 16 x 16 无符号乘法的指令序列。公式 7-1 给出所用的算法。最后得到的 32 位结果存储在 4 个寄存器（RES3:RES0）中。

公式 7-1： 16 x 16 无符号乘法算法

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

例 7-3： 16 x 16 无符号乘法程序

```
    MOVF    ARG1L, W
    MULWF   ARG2L         ; ARG1L * ARG2L ->
                          ; PRODH:PRODL
    MOVFF   PRODH,  RES1  ;
    MOVFF   PRODH,  RES1  ;
;
    MOVF    ARG1H,  W
    MULWF   ARG2H         ; ARG1H * ARG2H ->
                          ; PRODH:PRODL
    MOVFF   PRODH,  RES3  ;
    MOVFF   PRODL,  RES2  ;
;
    MOVF    ARG1L,  W
    MULWF   ARG2H         ; ARG1L * ARG2H ->
                          ; PRODH:PRODL
    MOVF    PRODL,  W     ;
    ADDWF   RES1,   F     ; Add cross
    MOVF    PRODH,  W     ; products
    ADDWFC  RES2,   F     ;
    CLRF    WREG          ;
    ADDWFC  RES3,   F     ;
;
    MOVF    ARG1H,  W     ;
    MULWF   ARG2L         ; ARG1H * ARG2L ->
                          ; PRODH:PRODL
    MOVF    PRODL,  W     ;
    ADDWF   RES1,   F     ; Add cross
    MOVF    PRODH,  W     ; products
    ADDWFC  RES2,   F     ;
    CLRF    WREG          ;
    ADDWFC  RES3,   F     ;
```

例 7-4 给出 16 x 16 有符号乘法的指令序列。公式 7-2 给出所用的算法。最后得到的 32 位结果存储在 4 个寄存器 （RES3:RES0）中。要考虑自变量的符号位，就要测试每个自变量对的最高有效位 （MSb），并执行正确的减法运算。

公式 7-2： 16 x 16 有符号乘法算法

RES3:RES0

$$
\begin{aligned}
&= ARG1H{:}ARG1L \cdot ARG2H{:}ARG2L \\
&= (ARG1H \cdot ARG2H \cdot 2^{16}) + \\
&\quad (ARG1H \cdot ARG2L \cdot 2^{8}) + \\
&\quad (ARG1L \cdot ARG2H \cdot 2^{8}) + \\
&\quad (ARG1L \cdot ARG2L) + \\
&\quad (-1 \cdot ARG2H{<}7{>} \cdot ARG1H{:}ARG1L \cdot 2^{16}) \\
&+
\end{aligned}
$$

例 7-4： 16 x 16 有符号的乘法程序

```
    MOVF    ARG1L,  W
    MULWF   ARG2L         ; ARG1L * ARG2L ->
                          ; PRODH:PRODL
    MOVFF   PRODH,  RES1  ;
    MOVFF   PRODH,  RES1  ;
;
    MOVF    ARG1H,  W
    MULWF   ARG2H         ; ARG1H * ARG2H ->
                          ; PRODH:PRODL
    MOVFF   PRODH,  RES3  ;
    MOVFF   PRODL,  RES2  ;
;
    MOVF    ARG1L,  W
    MULWF   ARG2H         ; ARG1L * ARG2H ->
                          ; PRODH:PRODL
    MOVF    PRODL,  W     ;
    ADDWF   RES1,   F     ; Add cross
    MOVF    PRODH,  W     ; products
    ADDWFC  RES2,   F     ;
    CLRF    WREG          ;
    ADDWFC  RES3,   F     ;
;
    MOVF    ARG1H,  W     ;
    MULWF   ARG2L         ; ARG1H * ARG2L ->
                          ; PRODH:PRODL
    MOVF    PRODL,  W     ;
    ADDWF   RES1,   F     ; Add cross
    MOVF    PRODH,  W     ; products
    ADDWFC  RES2,   F     ;
    CLRF    WREG          ;
    ADDWFC  RES3,   F     ;
;
    BTFSS   ARG2H,  7     ; ARG2H:ARG2L neg?
    BRA     SIGN_ARG1     ; no, check ARG1
    MOVF    ARG1H,  W     ;
    SUBWF   RES2          ;
    MOVF    ARG1H,  W     ;
    SUBWFB  RES3
;
SIGN_ARG1
    BTFSS   ARG2H,  7     ; ARG2H:ARG2L neg?
    BRA     CONT_CODE     ; no, done
    MOVF    ARG1H,  W     ;
    SUBWF   RES2          ;
    MOVF    ARG2H,  W     ;
    SUBWFB  RES3
;
CONT_CODE
      :
```


8.0 中断

PIC18FXX2 器件具有多个中断源及一个中断优先级功能，该功能可以给每个中断源分配高优先级中断或者低优先级中断。高优先级中断向量位于 000008h，低优先级中断向量位于 000018h。高优先级中断事件会覆盖掉任何正在执行的低优先级中断。

有 10 个寄存器用于控制中断操作。这些寄存器是：

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1 和 PIR2
- PIE1 和 PIE2
- IPR1 和 IPR2

建议这些寄存器中的符号位名称使用由MPLAB® IDE提供的 Microchip 头文件。这样汇编器 / 编译器能够自动处理指定寄存器内这些位的位置。

每个中断源 （INT0 除外）有 3 个位来控制其操作。这些位的功能是：

- 标志位，表明发生了中断事件
- 使能位，当该标志位置 1 时，允许程序执行转移到中断向量地址
- 优先级位，用于选择高优先级还是低优先级

通过将 IPEN 位 （RCON<7>）置 1，可使能中断优先级功能。使能中断优先级后，有2位可以使能全局中断。将 GIEH 位 （INTCON<7>）置 1，将使能所有已经置位优先级位的中断。将 GIEL 位 （INTCON<6>）置 1，将使能所有已经清零优先级位的中断。当中断标志位、使能位以及相应的全局中断使能位均被置 1 时，中断将根据设置的优先级立即转到地址 000008h 或 000018h。也可以通过设置相应的使能位来禁止单个中断。

当 IPEN 位清零 （缺省状态）时，便会禁止中断优先级功能，并且中断与 PICmicro® 中档系列器件兼容。在兼容模式下，各个中断源的中断优先级位均不起作用。INTCON<6> 是 PEIE 位，用于使能 / 禁止所有的外部中断源。 INTCON<7> 是 GIE 位，用于使能 / 禁止所有中断源。在 兼 容 模 式 下，所 有 中 断 均 跳 转 到 地 址 000008h。

当响应中断时， 全局中断使能位被清零以禁止其他中断。如果 IPEN 位是清零状态，这里即指 GIE 位。如果使用了中断优先级，则指 GIEH 位或者 GIEL 位。高优先级中断源可以中断低优先级中断。

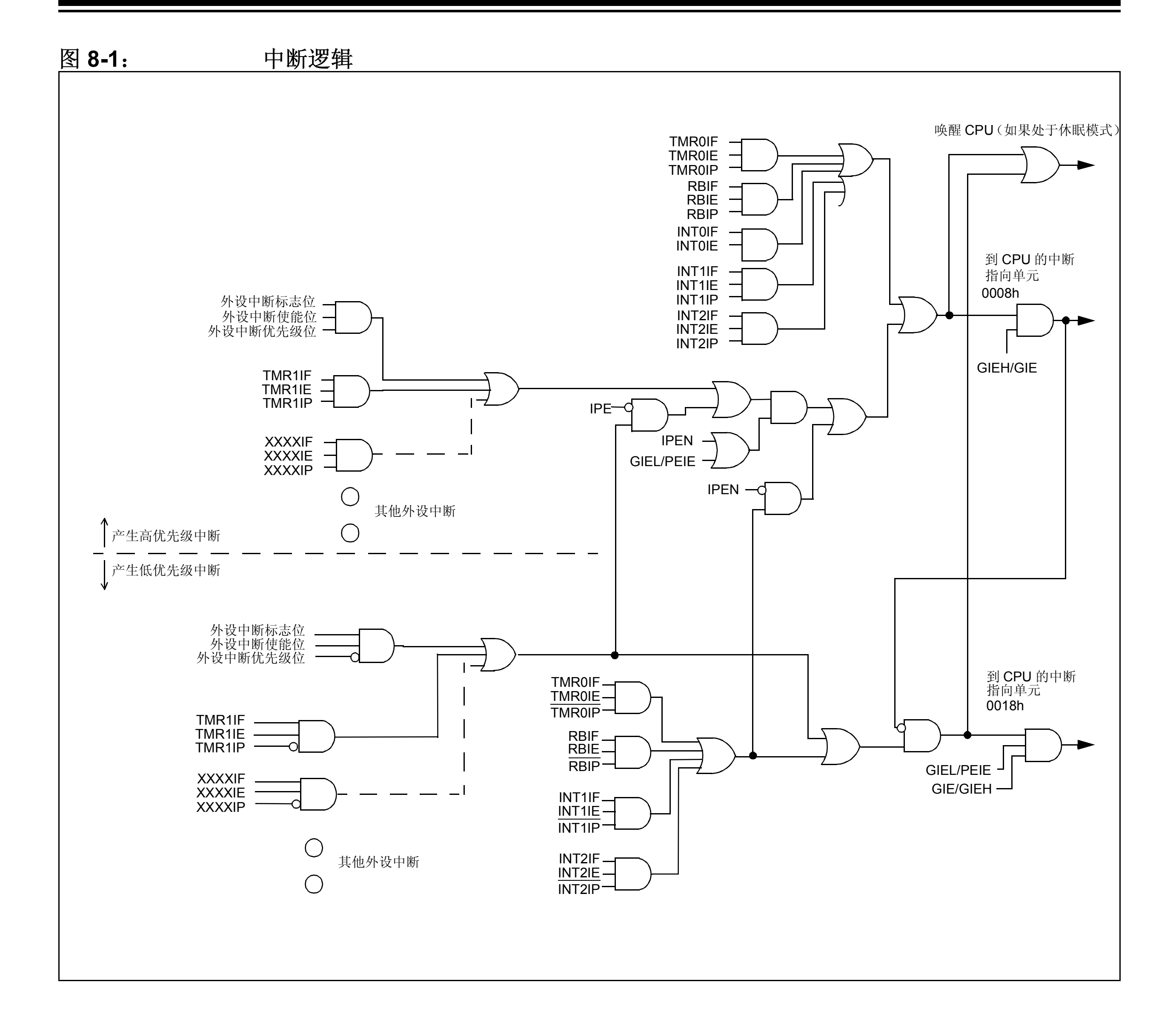
返回地址压入堆栈，将中断向量地址 （000008h 或 000018h）装入 PC。只要在中断服务程序中，就可以通过轮询中断标志位来确定中断源。中断标志位必须在重新使能中断之前用软件清零，以避免重复响应这些中断。

执行 “中断返回” 指令 `RETFIE` 将退出中断程序，同时将 GIE 位 （如果使用优先级则为 GIEH 位或 GIEL 位）置 1，从而重新使能中断。

对于外部中断事件，例如 INT 引脚中断或者 PORTB 输入电平变化中断，中断响应延时将是3到4个指令周期。对于单周期或双周期指令，中断响应延时完全相同。各中断标志位的置位不受对应的中断使能位或GIE位状态的影响。

注：	当**任何**中断被使能时，不要使用 `MOVFF` 指令修改任何中断控制寄存器。否则可能导致单片机操作出错。

图 8-1： 中断逻辑


唤醒 CPU（如果处于休眠模式）
TMR0IF
TMR0IE
TMR0IP
RBIF
RBIE
RBIP
INT0IF
INT0IE
INT1IF
INT1IE
INT1IP
INT2IF
INT2IE
INT2IP
到 CPU 的中断
指向单元
0008h
GIEH/GIE
外设中断标志位
外设中断使能位
外设中断优先级位
TMR1IF
TMR1IE
TMR1IP
XXXXIF
XXXXIE
XXXXIP
IPE
IPEN
GIEL/PEIE
IPEN
其他外设中断
产生高优先级中断
产生低优先级中断
外设中断标志位
外设中断使能位
外设中断优先级位
TMR1IF
TMR1IE
TMR1IP
XXXXIF
XXXXIE
XXXXIP
其他外设中断
TMR0IF
TMR0IE
TMR0IP
RBIF
RBIE
RBIP
INT1IF
INT1IE
INT1IP
INT2IF
INT2IE
INT2IP
到 CPU 的中断
指向单元
0018h
GIEL/PEIE
GIE/GIEH

8.1 INTCON 寄存器

INTCON 寄存器是可读写寄存器，包含了多个使能位、优先级位和标志位。

> **注：** 当发生中断时，不管相应的中断使能位或全局使能位状态如何，中断标志位都将置 1。用户软件应在使能一个中断之前，先将相应的中断标志位清零。故此功能允许进行软件轮询。

寄存器 8-1： INTCON 寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

bit 7 **GIE/GIEH**：全局中断使能位
当 IPEN = 0 时：
1 =使能所有未被屏蔽的中断
0 =禁止所有中断
当 IPEN = 1 时：
1 =使能所有高优先级中断
0 =禁止所有中断

bit 6 **PEIE/GIEL**：外设中断使能位
当 IPEN = 0 时：
1 =使能所有未被屏蔽的外设中断
0 =禁止所有外设中断
当 IPEN = 1 时：
1 =使能所有低优先级的外设中断
0 =禁止所有低优先级的外设中断

bit 5 **TMR0IE**：TMR0 溢出中断使能位
1 =使能 TMR0 溢出中断
0 =禁止 TMR0 溢出中断

bit 4 **INT0IE**：INT0 外部引脚中断使能位
1 =使能 INT0 外部引脚中断
0 =禁止 INT0 外部引脚中断

bit 3 **RBIE**：RB 端口电平变化中断使能位
1 =使能 RB 端口电平变化中断
0 =禁止 RB 端口电平变化中断

bit 2 **TMR0IF**：TMR0 溢出中断标志位
1 = TMR0 寄存器已经溢出 （必须用软件清零）
0 = TMR0 寄存器没有溢出

bit 1 **INT0IF**：INT0 外部引脚中断标志位
1 =发生 INT0 外部中断 （必须用软件清零）
0 =未发生 INT0 外部中断

bit 0 **RBIF**：RB 端口电平变化中断标志位
1 = RB7:RB4 引脚中至少有一个引脚的电平状态发生变化 （必须用软件清零）
0 = RB7:RB4 引脚电平没有发生状态变化

注： 引脚上电平变化的情况会不断地将 RBIF 位置 1。而读取 PORTB 可以结束这种引脚电平变化的情况，并将 RBIF 位清零。

图注：			
R =可读位	W =可写位	U =未实现位，读为 0	
- n =上电复位时的值	1 =置位	0 =清零	x =未知位

寄存器 8-2： **INTCON2 寄存器**

R/W-1	R/W-1	R/W-1	R/W-1	U-0	R/W-1	U-0	R/W-1
$\overline{RBPU}$	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP
bit 7							bit 0

bit 7 **$\overline{RBPU}$**：PORTB 上拉使能位
1 ＝禁止所有 PORTB 上拉
0 ＝根据各端口锁存器值使能 PORTB 上拉

bit 6 **INTEDG0**：外部中断 0 触发边沿选择位
1 ＝上升沿触发中断
0 ＝下降沿触发中断

bit 5 **INTEDG1**：外部中断 1 触发边沿选择位
1 ＝上升沿触发中断
0 ＝下降沿触发中断

bit 4 **INTEDG2**：外部中断 2 触发边沿选择位
1 ＝上升沿触发中断
0 ＝下降沿触发中断

bit 3 **未实现**：读为 0

bit 2 **TMR0IP**：TMR0 溢出中断优先级位
1 ＝高优先级
0 ＝低优先级

bit 1 **未实现**：读为 0

bit 0 **RBIP**：RB 端口电平变化中断优先级位
1 ＝高优先级
0 ＝低优先级

图注：			
R ＝可读位	W ＝可写位	U ＝未实现位，读为 0	
- n ＝上电复位时的值	1 ＝置位	0 ＝清零	x ＝未知位

注： 当发生中断时，不管相应的中断使能位或全局使能位的状态如何，中断标志位都将置 1。用户软件应在使能一个中断之前，先将相应的中断标志位清零。故此功能允许进行软件轮询。

寄存器 8-3： INTCON3 寄存器

R/W-1	R/W-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF
bit 7							bit 0

bit 7 **INT2IP**：INT2 外部引脚中断优先级位
1 =高优先级
0 =低优先级

bit 6 **INT1IP**：INT1 外部引脚中断优先级位
1 =高优先级
0 =低优先级

bit 5 **未实现**：读为 0

bit 4 **INT2IE**：INT2 外部引脚中断使能位
1 =使能 INT2 外部引脚中断
0 =禁止 INT2 外部引脚中断

bit 3 **INT1IE**：INT1 外部引脚中断使能位
1 =使能 INT1 外部引脚中断
0 =禁止 INT1 外部引脚中断

bit 2 **未实现**：读为 0

bit 1 **INT2IF**：INT2 外部引脚中断标志位
1 =发生 INT2 外部中断 （必须用软件清零）
0 =未发生 INT2 外部中断

bit 0 **INT1IF**：INT1 外部引脚中断标志位
1 =发生 INT1 外部中断 （必须用软件清零）
0 =未发生 INT1 外部中断

图注：			
R =可读位	W =可写位	U =未实现位，读为 0	
- n =上电复位时的值	1 =置位	0 =清零	x =未知位

注： 当发生中断时，不管相应的中断使能位或全局使能位的状态如何，中断标志位都将置 1。用户软件应在使能一个中断之前，先将相应的中断标志位清零。故此功能允许进行软件轮询。

8.2 PIR 寄存器

PIR 寄存器包含各外设中断的标志位。根据外设中断源的数量，有两个外设中断标志寄存器（PIR1 和 PIR2）。

> 注 1：当发生中断时，不管相应的中断使能位和全局使能位 GIE（INTCON<7>）的状态如何，中断标志位都将置 1。
>
> 2: 用户软件应在使能中断之前，先将相应的中断标志位清零；并在响应该中断后，也应该将该中断标志位清零。

寄存器 8-4： PIR1：外设中断请求 （标志）寄存器 1

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF(1)**：并行从动端口读 / 写中断标志位
1 =发生了读 / 写操作 （必须用软件清零）
0 =未发生读 / 写操作

bit 6 **ADIF**：A/D 转换器中断标志位
1 =完成 A/D 转换 （必须用软件清零）
0 =未完成 A/D 转换

bit 5 **RCIF**：USART 接收中断标志位
1 =USART 接收缓冲器 RCREG 满 （当读取 RCREG 时清零）
0 =USART 接收缓冲器为空

bit 4 **TXIF**：USART 发送中断标志位 （请参见第 16.0 节，了解 TXIF 功能的详细信息）
1 =USART 发送缓冲器 TXREG 为空 （当写入 TXREG 时清零）
0 =USART 发送缓冲器满

bit 3 **SSPIF**：主控同步串行口中断标志位
1 =完成发送 / 接收 （必须用软件清零）
0 =等待发送 / 接收

bit 2 **CCP1IF**：CCP1 中断标志位
捕捉模式：
1 =发生了 TMR1 寄存器捕捉 （必须用软件清零）
0 =未发生 TMR1 寄存器捕捉
比较模式：
1 =发生了 TMR1 寄存器的比较匹配 （必须用软件清零）
0 =未发生 TMR1 寄存器的比较匹配
PWM 模式：
此模式下未使用

bit 1 **TMR2IF**：TMR2 对 PR2 匹配中断标志位
1 =TMR2 对 PR2 匹配 （必须用软件清零）
0 =TMR2 对 PR2 不匹配

bit 0 **TMR1IF**：TMR1 溢出中断标志位
1 =TMR1 寄存器发生溢出 （必须用软件清零）
0 =TMR1 寄存器没有溢出

注 1：在 PIC18F2X2 器件中， PSPIF 位是保留位；其值始终为 0。

图注：			
R =可读位	W =可写位	U =未实现位，读为 0	
- n =上电复位时的值	1 =置位	0 =清零	x =未知位

寄存器 8-5： **PIR2：外设中断请求 （标志）寄存器 2**

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF
bit 7							bit 0

bit 7-5 未实现：读为 0

bit 4 **EEIF**：数据 EEPROM/ 闪存写操作中断标志位
1 =完成写操作 （必须用软件清零）
0 =写操作未完成，或尚未开始

bit 3 **BCLIF**：总线冲突中断标志位
1 =发生了总线冲突 （必须用软件清零）
0 =未发生总线冲突

bit 2 **LVDIF**：低压检测中断标志位
1 =发生低电压状态 （必须用软件清零）
0 =器件的电压高于低压检测跳变点

bit 1 **TMR3IF**：TMR3 溢出中断标志位
1 =TMR3 寄存器溢出 （必须用软件清零）
0 =TMR3 寄存器没有溢出

bit 0 **CCP2IF**：CCPx 中断标志位
<u>输入捕捉模式：</u>
1 =发生了 TMR1 寄存器捕捉 （必须用软件清零）
0 =未发生 TMR1 寄存器捕捉
<u>输出比较模式：</u>
1 =发生了 TMR1 寄存器的比较匹配 （必须用软件清零）
0 =未发生 TMR1 寄存器的比较匹配
<u>PWM 模式：</u>
此模式下未使用

图注：			
R =可读位	W =可写位	U =未实现位，读为 0	
- n =上电复位时的值	1 =置位	0 =清零	x =未知位

8.3 PIE 寄存器

PIE 寄存器包含各外设中断的使能位。根据外设中断源的数量，可以有两个外设中断使能寄存器（PIE1 和 PIE2）。当 IPEN = 0 时，要使能任何一个外设中断，必须将 PEIE 位置 1。

寄存器 8-6： PIE1：外设中断使能寄存器 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE(1)**：并行从动端口读 / 写中断使能位
1 = 使能 PSP 的读 / 写中断
0 = 禁止 PSP 的读 / 写中断

bit 6 **ADIE**：A/D 转换器中断使能位
1 = 使能 A/D 中断
0 = 禁止 A/D 中断

bit 5 **RCIE**：USART 接收中断使能位
1 = 使能 USART 接收中断
0 = 禁止 USART 接收中断

bit 4 **TXIE**：USART 发送中断使能位
1 = 使能 USART 发送中断
0 = 禁止 USART 发送中断

bit 3 **SSPIE**：主控同步串行口中断使能位
1 = 使能 MSSP 中断
0 = 禁止 MSSP 中断

bit 2 **CCP1IE**：CCP1 中断使能位
1 = 使能 CCP1 中断
0 = 禁止 CCP1 中断

bit 1 **TMR2IE**：TMR2 对 PR2 匹配中断使能位
1 = 使能 TMR2 对 PR2 匹配中断
0 = 禁止 TMR2 对 PR2 匹配中断

bit 0 **TMR1IE**：TMR1 溢出中断使能位
1 = 使能 TMR1 溢出中断
0 = 禁止 TMR1 溢出中断

注 **1**：在 PIC18F2X2 器件中， PSPIE 位是保留位；其值始终为 0。

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
- n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知位

寄存器 8-7： PIE2：外设中断使能寄存器 2

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE
bit 7							bit 0

bit 7-5 **未实现：**读为 0

bit 4 **EEIE：**数据 EEPROM/ 闪存写操作中断使能位
1 =使能
0 =禁止

bit 3 **BCLIE：**总线冲突中断使能位
1 =使能
0 =禁止

bit 2 **LVDIE：**低压检测中断使能位
1 =使能
0 =禁止

bit 1 **TMR3IE：**TMR3 溢出中断使能位
1 =使能 TMR3 溢出中断
0 =禁止 TMR3 溢出中断

bit 0 **CCP2IE：**CCP2 中断使能位
1 =使能 CCP2 中断
0 =禁止 CCP2 中断

图注：			
R =可读位	W =可写位	U =未实现位，读为 0	
- n =上电复位时的值	1 =置位	0 =清零	x =未知位

8.4 IPR 寄存器

IPR 寄存器包含各外设中断的优先级位。根据外设中断源的数量，有两个外设中断优先级寄存器 （IPR1 和 IPR2）。对优先级位进行操作时，要求中断优先级使能（IPEN）位被置 1。

寄存器 8-8： IPR1：外设中断优先级寄存器 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PSPIP[(1)]	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

bit 7 **PSPIP[(1)]**：并行从动端口读 / 写中断优先级位
1 =高优先级
0 =低优先级

bit 6 **ADIP**：A/D 转换器中断优先级位
1 =高优先级
0 =低优先级

bit 5 **RCIP**：USART 接收中断优先级位
1 =高优先级
0 =低优先级

bit 4 **TXIP**：USART 发送中断优先级位
1 =高优先级
0 =低优先级

bit 3 **SSPIP**：主控同步串行口中断优先级位
1 =高优先级
0 =低优先级

bit 2 **CCP1IP**：CCP1 中断优先级位
1 =高优先级
0 =低优先级

bit 1 **TMR2IP**：TMR2 对 PR2 匹配中断优先级位
1 =高优先级
0 =低优先级

bit 0 **TMR1IP**：TMR1 溢出中断优先级位
1 =高优先级
0 =低优先级

注 **1**：在 PIC18F2X2 器件中， PSPIP 位是保留位；其值始终为 1。

图注：			
R =可读位	W =可写位	U =未实现位，读为 0	
- n =上电复位时的值	1 =置位	0 =清零	x =未知位

寄存器 8-9： IPR2：外设中断优先级寄存器 2

U-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP
bit 7							bit 0

bit 7-5 **未实现**：读为 0

bit 4 **EEIP**：数据 EEPROM/ 闪存写操作中断优先级位
1 =高优先级
0 =低优先级

bit 3 **BCLIP**：总线冲突中断优先级位
1 =高优先级
0 =低优先级

bit 2 **LVDIP**：低压检测中断优先级位
1 =高优先级
0 =低优先级

bit 1 **TMR3IP**：TMR3 溢出中断优先级位
1 =高优先级
0 =低优先级

bit 0 **CCP2IP**：CCP2 中断优先级位
1 =高优先级
0 =低优先级

图注：			
R =可读位	W =可写位	U =未实现位，读为 0	
- n =上电复位时的值	1 =置位	0 =清零	x =未知位

8.5 RCON 寄存器

RCON 寄存器包含可使能优先级中断的位 （IPEN）。

寄存器 8-10： RCON 寄存器

R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7 **IPEN：**中断优先级使能位
1 = 使能中断优先级
0 = 禁止中断优先级 （16CXXX 兼容模式）

bit 6-5 **未实现：**读为 0

bit 4 **$\overline{RI}$：**RESET 指令标志位
请参见寄存器 4-3，了解位操作的详细信息

bit 3 **$\overline{TO}$：**看门狗超时溢出标志位
请参见寄存器 4-3，了解位操作的详细信息

bit 2 **$\overline{PD}$：**掉电检测标志位
请参见寄存器 4-3，了解位操作的详细信息

bit 1 **$\overline{POR}$：**上电复位状态位
请参见寄存器 4-3，了解位操作的详细信息

bit 0 **$\overline{BOR}$：**欠压复位状态位
请参见寄存器 4-3，了解位操作的详细信息

图注：			
R =可读位	W =可写位	U =未实现位，读为 0	
- n =上电复位时的值	1 =置位	0 =清零	x =未知位

8.6 INT0 中断

RB0/INT0、RB1/INT1 及 RB2/INT2 引脚的外部中断是边沿触发的：如果 INTCON2 寄存器中相应的 INTEDGx 位被置 1，则为上升沿触发；如果该 INTEDGx 位清零，则为下降沿触发。当 RBx/INTx 引脚上出现一个有效边沿时，相应标志位 INTxF 被置 1。通过对相应的使能位 INTxE 清零，可以禁止该中断。在重新使能该中断前，必须在中断服务程序中先用软件将标志位 INTxF 清零。如果 INTxE 位在进入休眠状态前被置 1，则所有的外部中断（INT0、INT1 及 INT2）能把处理器从休眠状态中唤醒。如果全局中断使能位 GIE 被置 1，则处理器将在唤醒之后转移到中断向量。

INT1 和 INT2 的中断优先级由中断优先级位 INT1IP（INTCON3<6>）和 INT2IP（INTCON3<7>）中的值决定。没有与 INT0 有关的优先级位。INT0 始终是一个高优先级的中断源。

8.7 TMR0 中断

在 8 位模式（缺省模式）下，TMR0 寄存器的溢出（FFh → 00h）将会使标志位 TMR0IF 置 1。在 16 位模式下，TMR0H:TMR0L 寄存器的溢出（FFFFh → 0000h）将会使标志位 TMR0IF 置 1。通过对使能位 T0IE（INTCON<5>）置 1/ 清零，可以使能 / 禁止该中断。Timer0 的中断优先级由中断优先级位 TMR0IP（INTCON2<2>）中的值决定。欲进一步了解 Timer0 模块的详细内容，请参见第 10.0 节。

8.8 PORTB 电平变化中断

PORTB<7:4> 上的输入电平变化，会将标志位 RBIF（INTCON<0>）置 1。通过对使能位 RBIE（INTCON<3>）置 1/ 清零，可以使能 / 禁止该中断。PORTB 电平变化中断的中断优先级由中断优先级位 RBIP（INTCON2<0>）中的值决定。

8.9 中断的现场保护

在中断期间，将返回的 PC 值保存到堆栈。另外，将 WREG、STATUS 和 BSR 寄存器的值压入快速返回堆栈。如果未使用从中断快速返回（请参见第 4.3 节），那么用户可能需要用软件保存 WREG、STATUS 和 BSR 寄存器中的值。根据用户的具体应用，还可能需要保存其他寄存器的值。例 8-1 在执行中断服务程序期间，保存并恢复 WREG、STATUS 和 BSR 寄存器的值。

例 8-1： 在 RAM 内保存 STATUS、WREG 和 BSR 寄存器

```
MOVWF   W_TEMP                          ; W_TEMP is in virtual bank
MOVFF   STATUS, STATUS_TEMP             ; STATUS_TEMP located anywhere
MOVFF   BSR,    BSR_TEMP                ; BSR located anywhere
;
; USER ISR CODE
;
MOVFF   BSR_TEMP,   BSR                 ; Restore BSR
MOVF    W_TEMP,     W                   ; Restore WREG
MOVFF   STATUS_TEMP,STATUS              ; Restore STATUS
```

注：

9.0 I/O 端口

根据所选择的器件，有三个或五个端口可供选择。某些 I/O 端口的引脚用来与器件上外设元件的其他功能复用。一般来说，当相应的外设使能时，其对应的引脚不能作为通用 I/O 引脚使用。

每个端口有三个用于操作的寄存器。这些寄存器分别是：

- TRIS 寄存器 （数据方向寄存器）
- PORT 寄存器 （读取器件引脚的电平状态）
- LAT 寄存器 （输出锁存器）

在对 I/O 引脚电平驱动的值进行 “读—修改—写” 操作时会用到数据锁存器 （LAT 寄存器）。

例 9-1： 初始化 PORTA

```
CLRF  PORTA     ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRF  LATA      ; Alternate method
                ; to clear output
                ; data latches
MOVLW 0x07      ; Configure A/D
MOVWF ADCON1    ; for digital inputs
MOVLW 0xCF      ; Value used to
                ; initialize data
                ; direction
MOVWF TRISA     ; Set RA<3:0> as inputs
                ; RA<5:4> as outputs
```

9.1 PORTA、 TRISA 和 LATA 寄存器

PORTA 是一个 7 位宽的双向端口。对应的数据方向寄存器是 TRISA。当 TRISA 某位置 1 时，其相应 PORTA 引脚作为输入 （即，相应的输出驱动呈高阻态） 。如 TRISA 某位清零，则相应的 PORTA 引脚作为输出（即，输出锁存器中的数据从相应引脚输出）。

读 PORTA 寄存器是读取引脚上的电平状态，而写 PORTA 寄存器是将数据写入端口的数据锁存器。

数据锁存器（LATA）也是存储器映射的。对 LATA 寄存器的 “读－修改－写” 操作是读写锁存的 PORTA 的输出值。

RA4引脚与Timer0模块时钟输入复用，成为RA4/T0CKI 引脚。RA4/T0CKI 引脚是施密特触发器的输入和漏极开路输出。而其他的 RA 端口引脚都是 TTL 逻辑电平输入和完全 CMOS 驱动输出。

其他的 PORTA 引脚都可与模拟输入信号和模拟 VREF+ 及 VREF- 输入复用。通过对 ADCON1 寄存器 （A/D 控制寄存器1）的控制位清零/置1来选择各引脚的操作。

注：	在上电复位时，RA5 和 RA3:RA0 设置为模拟输入并读作 0。 RA6 和 RA4 设置为数字输入。

即使将 RA 引脚用于模拟输入，它们的输入输出方向仍然由 TRISA 寄存器控制。当 TRISA 寄存器中的某些位用作模拟输入时，用户必须确保这些位置 1。

图 9-1： RA3:RA0 和 RA5 引脚的结构图

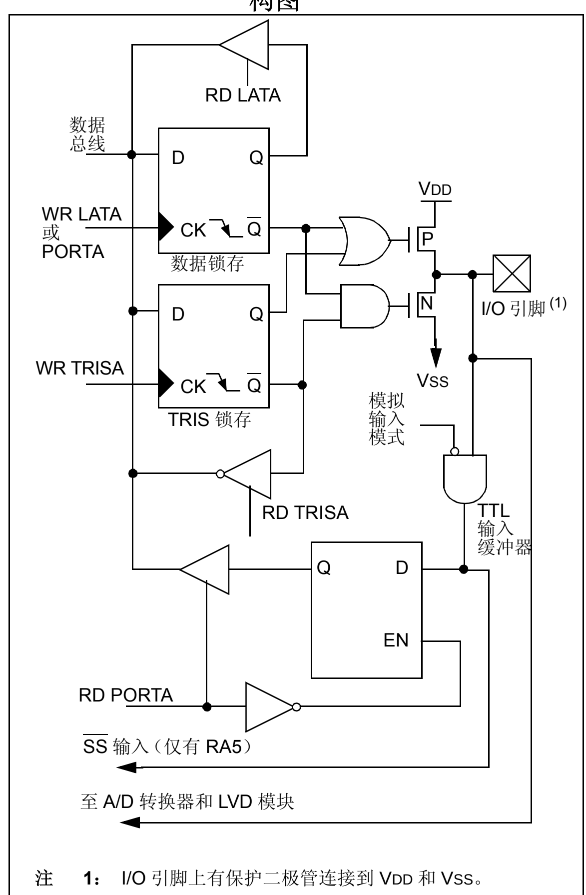
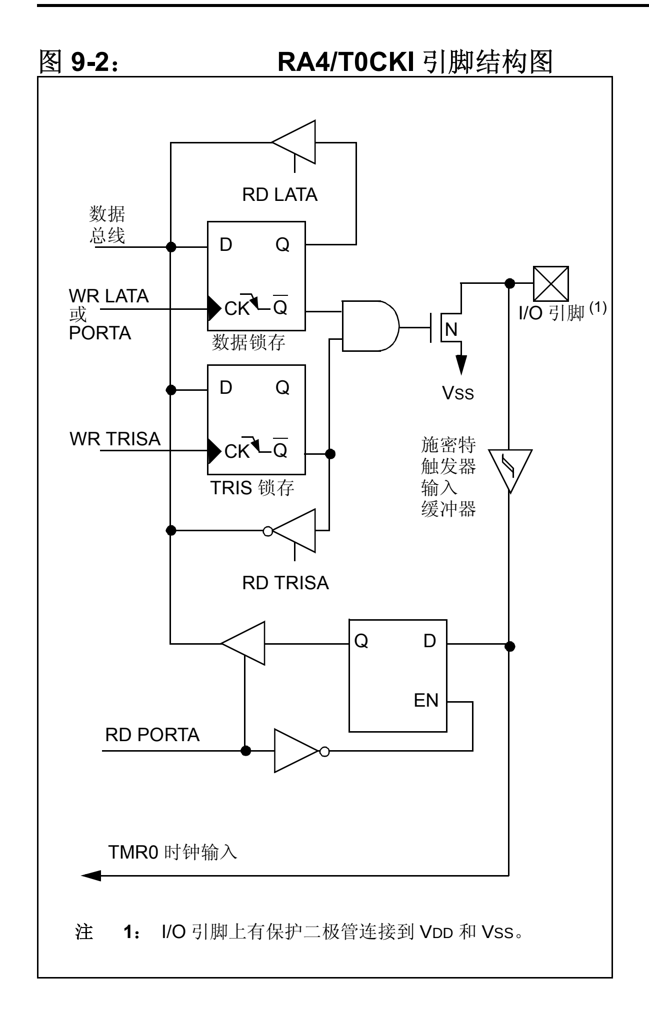
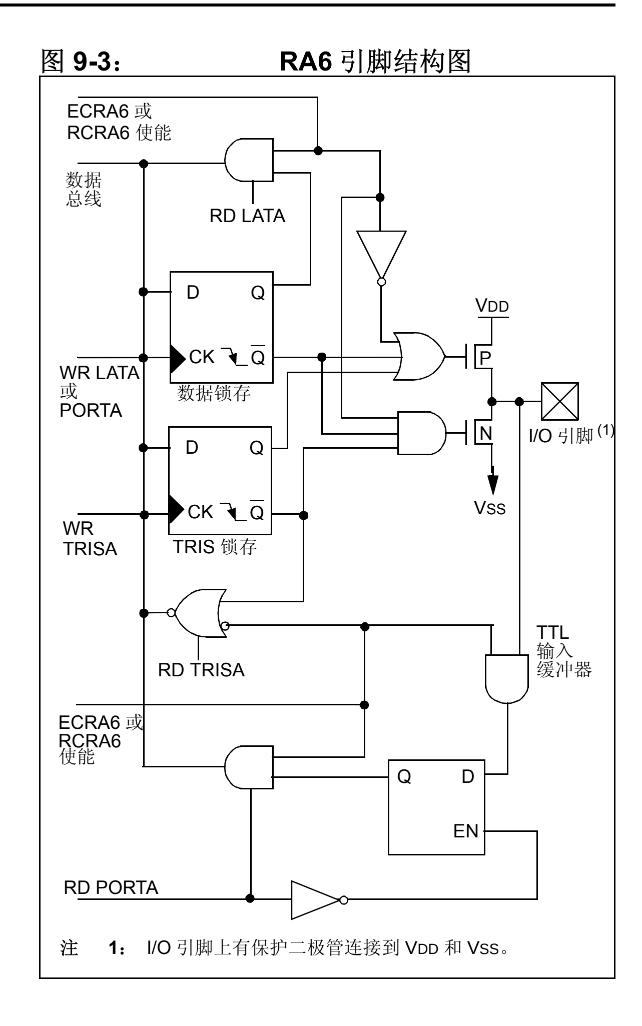


图 9-2： RA4/T0CKI 引脚结构图



注 1： I/O 引脚上有保护二极管连接到 VDD 和 VSS。

图 9-3： RA6 引脚结构图



注 1： I/O 引脚上有保护二极管连接到 VDD 和 VSS。

表 **9-1**： **PORTA** 引脚功能

名称	Bit#	缓冲器	功能
RA0/AN0	bit0	TTL	输入 / 输出或模拟输入。
RA1/AN1	bit1	TTL	输入 / 输出或模拟输入。
RA2/AN2/VREF-	bit2	TTL	输入 / 输出或模拟输入或 VREF-。
RA3/AN3/VREF+	bit3	TTL	输入 / 输出或模拟输入或 VREF+。
RA4/T0CKI	bit4	ST	输入 / 输出或 Timer0 的外部时钟输入。 输出是漏极开路型。
RA5/$\overline{SS}$/AN4/LVDIN	bit5	TTL	输入 / 输出、同步串行口的从动选择输入、模拟输入或低压检测输入。
OSC2/CLKO/RA6	bit6	TTL	OSC2、时钟输出或 I/O 引脚。

图注： TTL=TTL 输入， ST= 施密特触发输入

表 **9-2**： 与 **PORTA** 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
PORTA	—	RA6	RA5	RA4	RA3	RA2	RA1	RA0	-x0x 0000	-u0u 0000
LATA	—	LATA 数据输出寄存器							-xxx xxxx	-uuu uuuu
TRISA	—	PORTA 数据方向寄存器							-111 1111	-111 1111
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

图注： x = 未知，u = 不变， - = 未实现位，读为 0。阴影单元表示 PORTA 未使用。

9.2 PORTB、TRISB 和 LATB 寄存器

PORTB 是一个 8 位宽的双向端口。对应的数据方向寄存器是 TRISB。当 TRISB 某位置 1 时，其相应 PORTB 引脚作为输入（即，相应的输出驱动呈高阻态）。当 TRISB 某位清零时，则相应的 PORTB 引脚作为输出（即，输出锁存器中的数据从相应引脚输出）。

数据锁存器（LATB）也是存储器映射的。对 LATB 寄存器的“读—修改—写”操作是读写锁存的 PORTB 的输出值。

例 9-2： 初始化 PORTB

```
CLRF   PORTB   ; Initialize PORTB by
               ; clearing output
               ; data latches
CLRF   LATB    ; Alternate method
               ; to clear output
               ; data latches
MOVLW 0xCF     ; Value used to
               ; initialize data
               ; direction
MOVWF TRISB    ; Set RB<3:0> as inputs
               ; RB<5:4> as outputs
               ; RB<7:6> as inputs
```

PORTB的每个引脚都有内部弱上拉电阻。只要将$\overline{\text{RBPU}}$位（INTCON2<7>）清零，就可以开启所有的上拉电阻。当端口引脚设置为输出时，其弱上拉电阻会自动断开。在上电复位后，会禁止弱上拉电阻。

> **注：** 当上电复位时，这些引脚被配置为数字输入。

PORTB 的四个引脚（RB7:RB4）具有电平变化中断功能。仅在这些引脚被设置为输入时，才可能发生此中断（即，当 RB7:RB4 的任何一个引脚被设置为输出时，该引脚不再具有电平变化中断功能）。将当前 RB7:RB4 引脚上的输入电平与前次从 PORTB 读入锁存器的旧值进行比较。对 RB7:RB4 引脚上的电平变化输出进行“或”运算在 RBIF 标志位（INTCON<0>）上产生 RB 端口电平变化中断。

该中断可以将器件从休眠状态唤醒。在中断服务程序中，用户可以采用以下方式来清除中断：

a) 对 PORTB 进行读 / 写操作（但 MOVFF 指令例外）。这将结束电平变化情况。

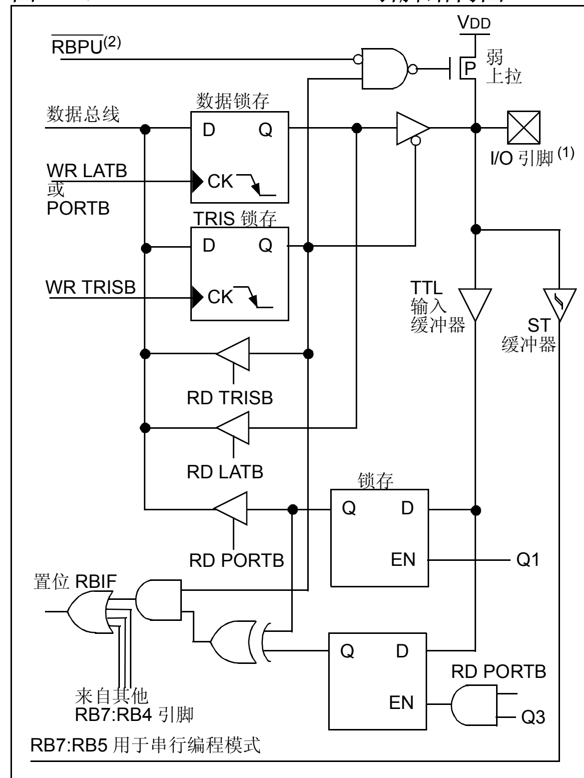
b) RBIF 标志位清零。

引脚上电平变化会不断地将 RBIF 标志位置 1。而对 PORTB 进行读操作，将结束引脚电平变化的情况，进而将 RBIF 标志位清零。

建议将电平变化中断功能用于按键唤醒操作及 PORTB 仅用来实现电平变化中断功能的操作情况。在使用电平变化中断功能时，不必对 PORTB 的状态进行轮询。

可以通过配置位 CCP2MX 将 RB3 设置为 CCP2 模块连接外设的引脚（CCP2MX='0'）。

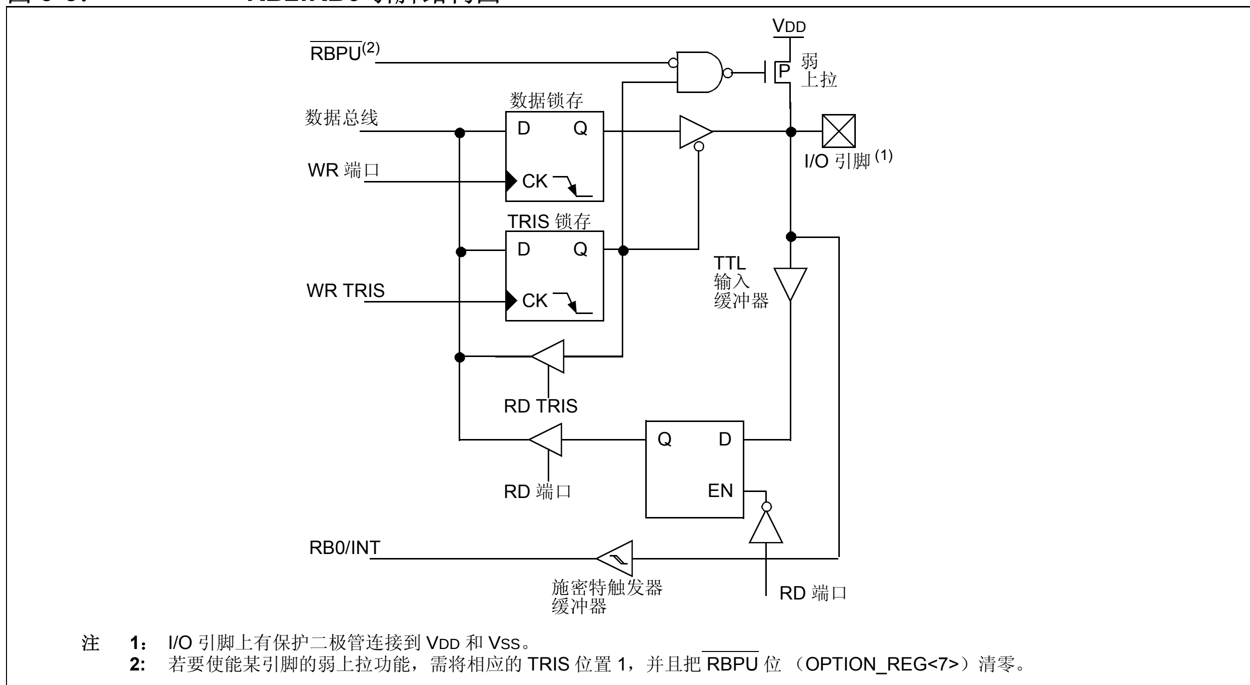
图 9-4： RB7:RB4 引脚结构图



注 1： I/O 引脚上有保护二极管连接到 VDD 和 VSS。
2： 若要使能某引脚的弱上拉功能，需将相应的 TRIS 位置 1，并且把 $\overline{\text{RBPU}}$ 位（INTCON2<7>）清零。

> **注 1：** 在低电压 ICSP 模式下，RB5 引脚不再用作通用 I/O 引脚，它在正常操作中应保持为低电平，以保护电路不受偶发性 ICSP 模式侵入。
>
> **2：** 当使用低电压 ICSP 编程（LVP）时，禁止 RB5 上拉电阻。如果 TRISB 的 bit5 被清零，就把 RB5 设置为输出，LATB 的 bit5 也必须被清零才能正确工作。


图 9-5： RB2:RB0 引脚结构图



注 1： I/O 引脚上有保护二极管连接到 VDD 和 VSS。
2： 若要使能某引脚的弱上拉功能，需将相应的 TRIS 位置 1，并且把 RBPU 位 （OPTION_REG<7>）清零。

图 9-6： RB3 引脚的结构图

VDD
RBPU(2)
CCP2MX
弱
上拉
P
CCP 输出 (3)
1
VDD
P
0
使能 (3)
CCP 输出
数据锁存
数据总线
D
Q
I/O 引脚 (1)
WR LATB 或
WR PORTB
CK
N
Vss
TRIS 锁存
D
TTL
输入
缓冲器
WR TRISB
CK
Q
RD TRISB
RD LATB
Q
D
RD PORTB
EN
RD PORTB
CCP2 输入 (3)
施密特触发器
缓冲器
CCP2MX = 0

注 1： I/O 引脚上有保护二极管连接到 VDD 和 VSS。
2： 若要使能某引脚的弱上拉功能，需将相应的 TRIS 位置 1，并且把 RBPU 位 （OPTION_REG<7>）清零。
3： 如果配置寄存器中 CCP2MX 位被使能 （为 0），则 CCP2 输入 / 输出与 RB3 复用。

表 9-3： PORTB 引脚功能

名称	Bit#	缓冲器	功能
RB0/INT0	bit0	TTL/ST(1)	输入 / 输出引脚或外部中断 input0。 可软件编程的内部弱上拉。
RB1/INT1	bit1	TTL/ST(1)	输入 / 输出引脚或外部中断 input1。 可软件编程的内部弱上拉。
RB2/INT2	bit2	TTL/ST(1)	输入 / 输出引脚或外部中断 input2。 可软件编程的内部弱上拉。
RB3/CCP2(3)	bit3	TTL/ST(4)	输入 / 输出引脚，当使能 CCP2MX 配置位时，为 Capture2 输入 / Capture2 输出 /PWM 输出。 可软件编程的内部弱上拉。
RB4	bit4	TTL	输入 / 输出引脚 （带电平变化中断功能）。 可软件编程的内部弱上拉。
RB5/PGM(5)	bit5	TTL/ST(2)	输入 / 输出引脚 （带电平变化中断功能）。 可软件编程的内部弱上拉。 低电压 ICSP 使能引脚。
RB6/PGC	bit6	TTL/ST(2)	输入 / 输出引脚 （带电平变化中断功能）。 可软件编程的内部弱上拉。 串行编程时钟。
RB7/PGD	bit7	TTL/ST(2)	输入 / 输出引脚 （带电平变化中断功能）。 可软件编程的内部弱上拉。 串行编程数据。

图注： TTL=TTL 输入， ST= 施密特触发输入

注 1：如果将该缓冲器配置为外部中断，为施密特触发输入。
2：如果该缓冲器在串行编程模式下使用，为施密特触发输入。
3：通过器件配置位选择与 CCP2 引脚复用的 I/O 引脚。
4：如果将该缓冲器配置为 CCP2 输入，为施密特触发输入。
5：在缺省情况下，低电平 ICSP 编程 （LVP）是使能的，而 RB5 I/O 功能则被禁止。如果要将 RB5 作为 I/O 引脚使能，并最大程度的兼容其他 28 引脚和 40 引脚的中档器件，就必须禁止 LVP。

表 9-4： 与 PORTB 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
LATB	LATB 数据输出寄存器								xxxx xxxx	uuuu uuuu
TRISB	PORTB 数据方向寄存器								1111 1111	1111 1111
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	—	TMR0IP	—	RBIP	1111 -1-1	1111 -1-1
INTCON3	INT2IP	INT1IP	—	INT2IE	INT1IE	—	INT2IF	INT1IF	11-0 0-00	11-0 0-00

图注： x = 未知，u = 不变。阴影单元表示 PORTB 未使用。

9.3 PORTC、 TRISC 和 LATC 寄存器

PORTC 是一个 8 位宽的双向端口。对应的数据方向寄存器是 TRISC。当 TRISC 某位置 1 时，其相应 PORTC 引脚作为输入（即，相应的输出驱动呈高阻态）。当 TRISC某位清零，则相应的PORTC引脚作为输出（即，输出锁存器中的数据从相应引脚输出）。

数据锁存器（LATC）也是存储器映射的。对 LATC 寄存器的“读—修改—写”操作是读写锁存的 PORTC 的输出值。

PORTC 与一些外设功能复用（表 9-5）。PORTC 引脚具有施密特触发输入缓冲器。

当使能外设功能时，应仔细定义每个 PORTC 引脚的 TRIS 位。有些外设使能时，会将相应引脚的 TRIS 位改为输出引脚；而另外一些外设使能时，会将相应引脚的 TRIS 位改为输入引脚。用户应该查阅相关的外设章节，以了解 TRIS 位的正确设置。

注：	当上电复位时，这些引脚被配置为数字输入。

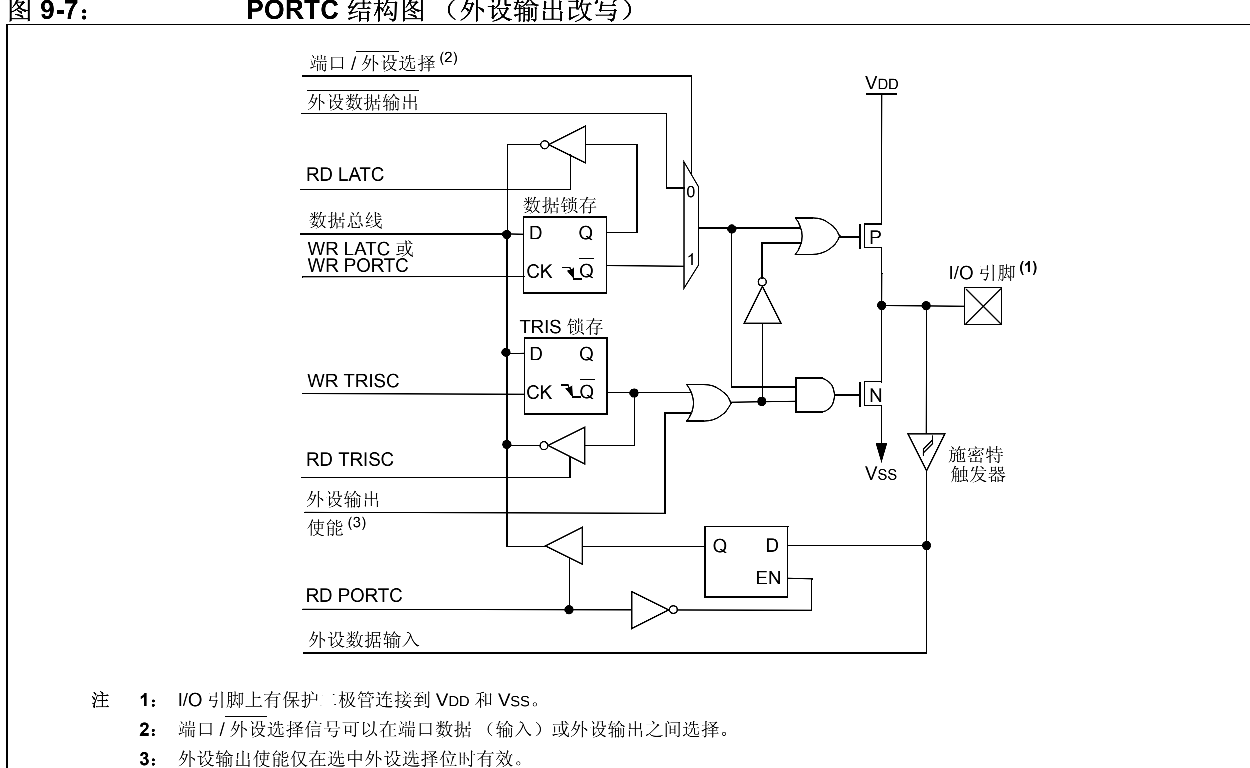
由于不会将改写引脚的值写入TRIS寄存器，因而在“读—修改—写”TRIS 寄存器时不必考虑外设改写的情况。

通常，通过配置位 CCP2MX 将 RC1 设置为 CCP2 模块的缺省外设引脚（缺省 / 擦除状态， CCP2MX = ’1’）。

例 9-3： 初始化 PORTC

```
CLRF   PORTC    ; Initialize PORTC by
                ; clearing output
                ; data latches
CLRF   LATC     ; Alternate method
                ; to clear output
                ; data latches
MOVLW 0xCF      ; Value used to
                ; initialize data
                ; direction
MOVWF TRISC     ; Set RC<3:0> as inputs
                ;RC<5:4> as outputs
                ; RC<7:6> as inputs
```

图 9-7： PORTC 结构图 （外设输出改写）



注 1： I/O 引脚上有保护二极管连接到 VDD 和 VSS。
2： 端口 / 外设选择信号可以在端口数据 （输入）或外设输出之间选择。
3： 外设输出使能仅在选中外设选择位时有效。

表 9-5： **PORTC** 引脚功能

名称	Bit#	缓冲器类型	功能
RC0/T1OSO/T1CKI	bit0	ST	输入 / 输出端口引脚或 Timer1 振荡器输出 /Timer1 时钟输入。
RC1/T1OSI/CCP2	bit1	ST	输入 / 输出端口引脚、 Timer1 振荡器输入或当 CCP2MX 配置位为 1 时为 Capture2 输入 /Compare2 输出 /PWM 输出。
RC2/CCP1	bit2	ST	输入 / 输出端口引脚或 Capture1 输入 /Compare1 输出 /PWM1 输出。
RC3/SCK/SCL	bit3	ST	RC3 也可以作为 SPI 和 I^{2}C 模式下的同步串行时钟。
RC4/SDI/SDA	bit4	ST	RC4 也可以作为 SPI 数据输入 （SPI 模式）或数据 I/O （I^{2}C 模式）。
RC5/SDO	bit5	ST	输入 / 输出端口引脚或同步串行口数据输出。
RC6/TX/CK	bit6	ST	输入 / 输出端口引脚、可寻址 USART 异步发送或可寻址 USART 同步时钟。
RC7/RX/DT	bit7	ST	输入 / 输出端口引脚、可寻址 USART 异步接收或可寻址 USART 同步数据。

图注： ST = 施密特触发器输入

表 9-6： 与 **PORTC** 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
LATC	LATC 数据输出寄存器								xxxx xxxx	uuuu uuuu
TRISC	PORTC 数据方向寄存器								1111 1111	1111 1111

图注： x = 未知，u = 不变

9.4 PORTD、 TRISD 和 LATD 寄存器

这部分内容仅适用于 PIC18F4X2 器件。

PORTD 是一个 8 位宽的双向端口。对应的数据方向寄存器是 TRISD。将 TRISD 位置 1，可使其相应 PORTD 引脚作为输入 （即，相应的输出驱动呈高阻态） 。将 TRISD 位清零，可使相应 PORTD 引脚作为输出（即，输出锁存器中的数据从相应引脚输出）。

数据锁存器 （LATD）也是存储器映射的。对 LATD 寄存器的 “读—修改—写”操作是读写锁存的 PORTD 的输出值。

PORTD是一个带有施密特触发输入缓冲器的8位端口。各引脚可分别配置为输入或输出。

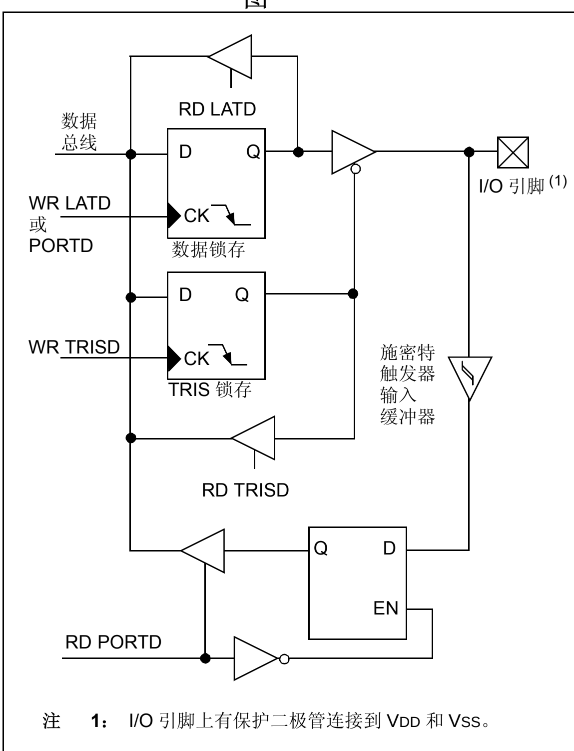
注：	当上电复位时，这些引脚被配置为数字输入。

通过将控制位 PSPMODE（TRISE<4>）置 1，PORTD 可以设置为 8 位宽的单片机端口 （并行从动端口）。在这种模式下，输入缓冲器是 TTL 输入。可以查阅第 9.6 节以获取有关并行从动端口 （PSP）的更多信息。

例 9-4： 初始化 PORTD

```
CLRF   PORTD  ; Initialize PORTD by
              ; clearing output
              ; data latches
CLRF   LATD   ; Alternate method
              ; to clear output
              ; data latches
MOVLW 0xCF    ; Value used to
              ; initialize data
              ; direction
MOVWF TRISD   ; Set RD<3:0> as inputs
              ; RD<5:4> as outputs
              ; RD<7:6> as inputs
```

图 9-8： I/O 口模式下的 PORTD 结构图



注 1： I/O 引脚上有保护二极管连接到 VDD 和 VSS。

表 9-7： **PORTD** 引脚功能

名称	**Bit#**	缓冲器类型	功能
RD0/PSP0	bit0	ST/TTL(1)	输入 / 输出端口引脚或并行从动端口 bit0。
RD1/PSP1	bit1	ST/TTL(1)	输入 / 输出端口引脚或并行从动端口 bit1。
RD2/PSP2	bit2	ST/TTL(1)	输入 / 输出端口引脚或并行从动端口 bit2。
RD3/PSP3	bit3	ST/TTL(1)	输入 / 输出端口引脚或并行从动端口 bit3。
RD4/PSP4	bit4	ST/TTL(1)	输入 / 输出端口引脚或并行从动端口 bit4。
RD5/PSP5	bit5	ST/TTL(1)	输入 / 输出端口引脚或并行从动端口 bit5。
RD6/PSP6	bit6	ST/TTL(1)	输入 / 输出端口引脚或并行从动端口 bit6。
RD7/PSP7	bit7	ST/TTL(1)	输入 / 输出端口引脚或并行从动端口 bit7。

图注： ST= 施密特触发输入， TTL=TTL 输入

注1：输入缓冲器在 I/O 模式下是施密特触发器输入缓冲器，而在并行从动端口模式下是 TTL 输入缓冲器。

表 9-8： 与 **PORTD** 有关的寄存器汇总

名称	**Bit 7**	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**	**POR 和 BOR 时的值**	**其他复位时的值**
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
LATD	LATD 数据输出寄存器								xxxx xxxx	uuuu uuuu
TRISD	PORTD 数据方向寄存器								1111 1111	1111 1111
TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE 数据方向位			0000 -111	0000 -111

图注： x = 未知，u = 不变， - = 未实现位，读为 0。阴影单元表示 PORTD 未使用。

9.5 PORTE、 TRISE 和 LATE 寄存器

本节内容仅适用于 PIC18F4X2 器件。

PORTE 是一个 3 位宽的双向端口。对应的数据方向寄存器是 TRISE。将 TRISE 位置 1，使相应的 PORTE 引脚作为输入（即，相应的输出驱动呈高阻态）。将TRISE 位清零，使相应的 PORTE 引脚作为输出 （即，输出锁存器中的数据从相应引脚输出）。

数据锁存器 （LATE）也是存储器映射的。对 LATE 寄存器的 “读—修改—写” 操作是读写锁存的 PORTE 输出值。

PORTE 有三个引脚（RE0/RD/AN5、RE1/WR/AN6 和 RE2/CS/AN7），可对其分别进行配置，或为输入或为输出。这些引脚都有施密特触发输入缓冲器。

寄存器 9-1 是 TRISE 寄存器，它控制并行从动端口操作。

PORTE 引脚与模拟输入复用。当选择引脚用作模拟输入时，这些引脚都读为 “0”。

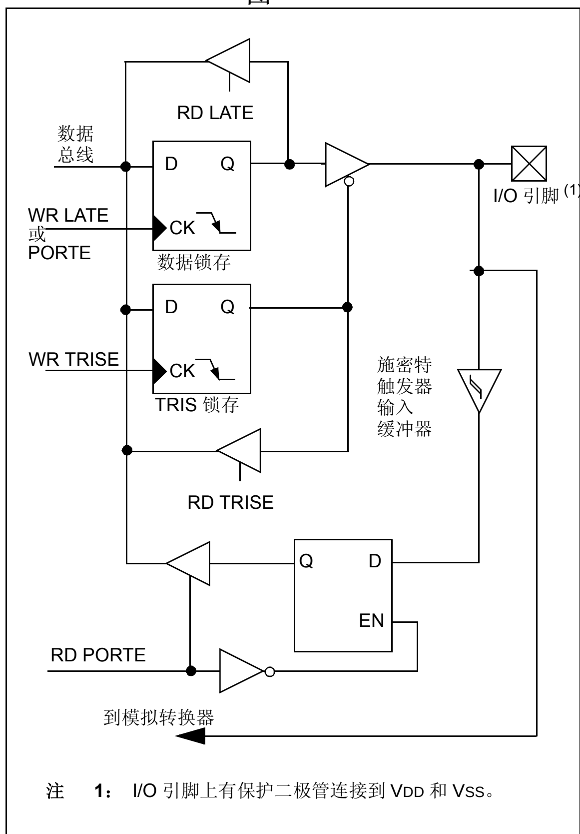
即使将 RE 引脚用作模拟输入，TRISE 寄存器仍将控制它们的输入输出方向。用户在使用 RE 引脚作为模拟输入时，应始终将其配置为输入引脚。

注：	当上电复位时，这些引脚被设置为模拟输入。

例 9-5： 初始化 PORTE

```
CLRF   PORTE   ;Initialize PORTE by
               ; clearing output
               ; data latches
CLRF   LATE    ; Alternate method
               ; to clear output
               ; data latches
MOVLW 0x07     ; Configure A/D
MOVWF ADCON1   ; for digital inputs
MOVLW 0x05     ;Value used to
               ; initialize data
               ; direction
MOVWF TRISE    ;Set RE<0> as inputs
               ;RE<1> as outputs
               ;RE<2> as inputs
```

图 9-9： I/O 口模式下的 PORTE 结构图



注 1： I/O 引脚上有保护二极管连接到 VDD 和 VSS。

寄存器 9-1： TRISE 寄存器

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0
bit 7							bit 0

bit 7 **IBF**：输入缓冲器满状态位
1 = 接收到一个字，等待 CPU 读取
0 = 未接收到任何字

bit 6 **OBF**：输出缓冲器满状态位
1 = 输出缓冲器仍保存着上一次写入的字
0 = 已读取输出缓冲器

bit 5 **IBOV**：输入缓冲器溢出检测位 （在微处理器模式下）
1 = 在尚未读取上一次输入的字前发生了一次写入
（该位必须用软件清零）
0 = 无溢出

bit 4 **PSPMODE**：并行从动端口模式选择位
1 = 并行从动端口模式
0 = 通用 I/O 模式

bit 3 **未实现**：读为 0

bit 2 **TRISE2**：RE2 方向控制位
1 = 输入
0 = 输出

bit 1 **TRISE1**：RE1 方向控制位
1 = 输入
0 = 输出

bit 0 **TRISE0**：RE0 方向控制位
1 = 输入
0 = 输出

图注：			
R = 可读位	W = 可写位	U = 未实现位，读作 0	
- n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知位

表 9-9： PORTE 引脚功能

名称	Bit#	缓冲器类型	功能
RE0/$\overline{RD}$/AN5	bit0	ST/TTL[1]	输入 / 输出端口引脚、并行从动端口模式下的读控制输入或模拟输入：$\overline{RD}$ 1 = 非读操作 0 = 读操作。读 PORTD 寄存器 （若片选有效）。
RE1/$\overline{WR}$/AN6	bit1	ST/TTL[1]	输入 / 输出端口引脚、并行从动端口模式下的写控制输入或模拟输入：$\overline{WR}$ 1 = 非写操作 0 = 写操作。写 PORTD 寄存器 （若片选有效）。
RE2/$\overline{CS}$/AN7	bit2	ST/TTL[1]	输入/输出端口引脚或并行从动端口模式下的片选控制输入或模拟输入：$\overline{CS}$ 1 = 未选中器件 0 = 选中器件

图注： ST= 施密特触发输入， TTL=TTL 输入

注1：输入缓冲器在 I/O 模式下是施密特触发输入缓冲器，而在并行从动端口模式下是 TTL 输入缓冲器。

表 9-10： 与 PORTE 相关的寄存器汇总

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -000	---- -000
LATE	—	—	—	—	—	LATE 数据输出寄存器			---- -xxx	---- -uuu
TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE 数据方向位			0000 -111	0000 -111
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

图注： x = 未知，u = 不变， - = 未实现位，读为 0。阴影单元表示 PORTE 未使用。

9.6 并行从动端口

并行从动端口仅适用于 40 引脚器件 （PIC18F4X2）。

当控制位 PSPMODE （TRISE<4>）置 1 时， PORTD 可作为一个 8 位宽的并行从动端口（即，微处理器端口）。通过 RD 控制输入引脚 RE0/RD 和 WR 控制输入引脚 RE1/WR，可以实现外部异步读写。

它可直接与 8 位微处理器数据总线相连。外部微处理器能够把 PORTD 锁存器当作 8 位锁存器，进行读写操作。将 PSPMODE 位置 1，使端口引脚 RE0/RD 成为 RD 输入，使 RE1/WR 成为 WR 输入，使 RE2/CS 成为 CS（片选）输入。对于这项功能，TRISE 寄存器的相关数据方向位（TRISE<2:0>）必须配置为输入（置 1）。A/D 端口配置位 PCFG2:PCFG0（ADCON1<2:0>）必须置 1，这样就把引脚 RE2:RE0 设置为数字 I/O。

首次检测 CS 和 WR 为低电平时，向 PSP 进行写入。首次检测 CS 和 RD 为低电平时，从 PSP 读取。

当 PSPMODE 位 （TRISE<4>）置 1 时， PORTE I/O 引脚成为微处理器端口的控制输入。在这种模式下，用户必须确保 TRISE<2:0> 置 1 （对应引脚配置为数字输入），并且 ADCON1 配置为数字 I/O。在这种模式下，输入缓冲器是 TTL 输入。

图 9-10： PORTD 和 PORTE 结构图 （并行从动端口）

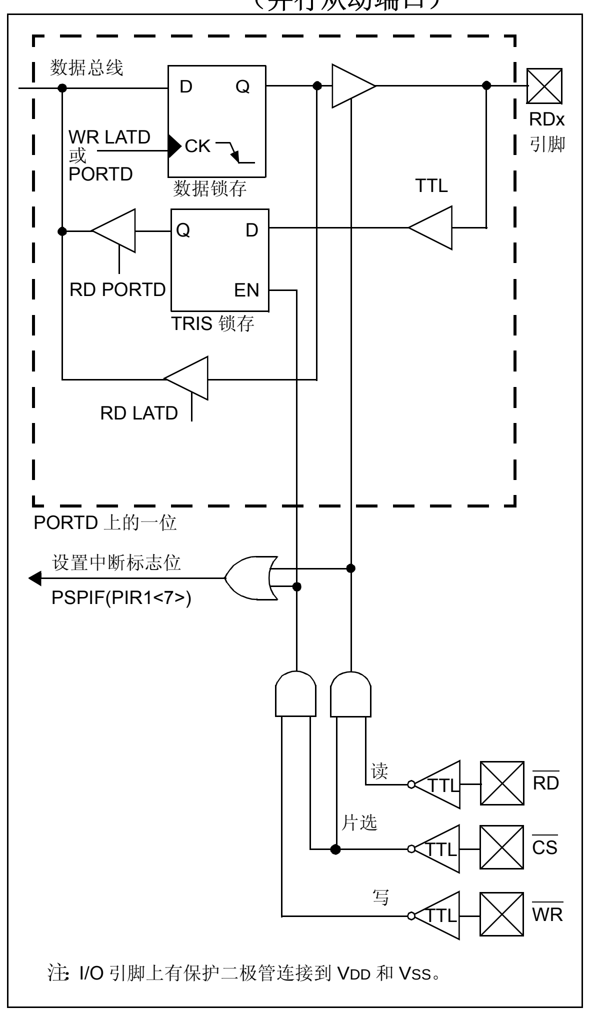


图 9-11： 并行从动端口写操作波形图

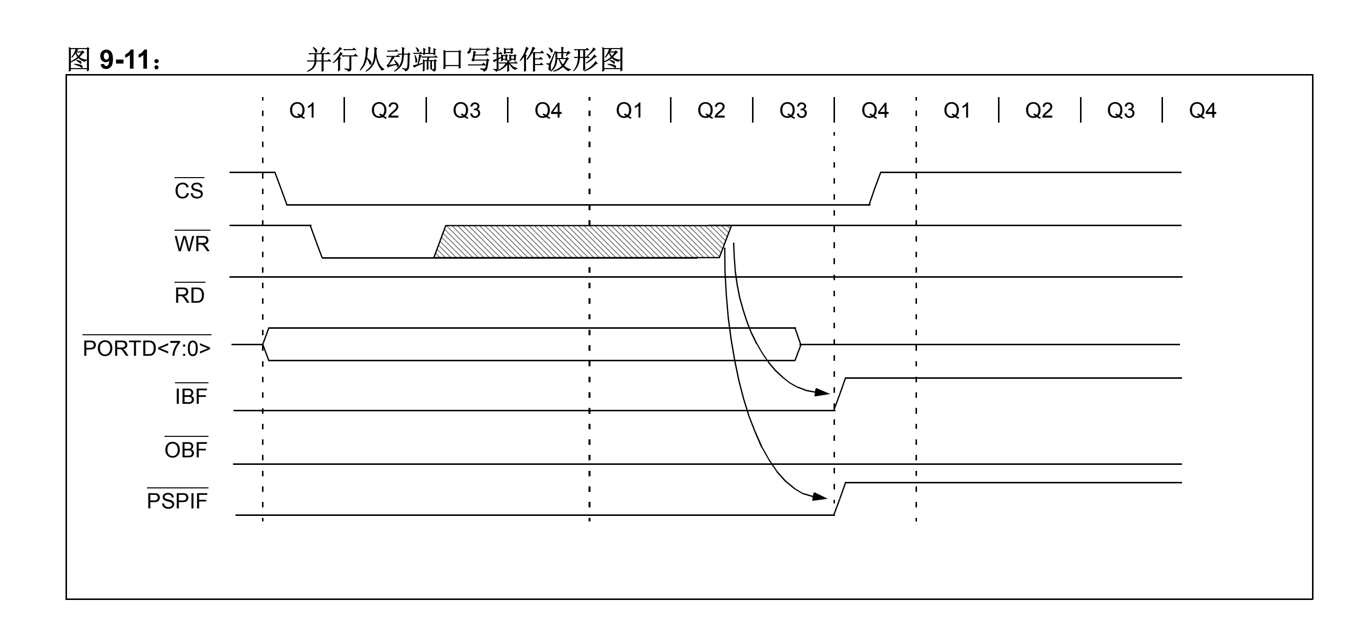


图 **9-12：** 并行从动端口读操作波形图

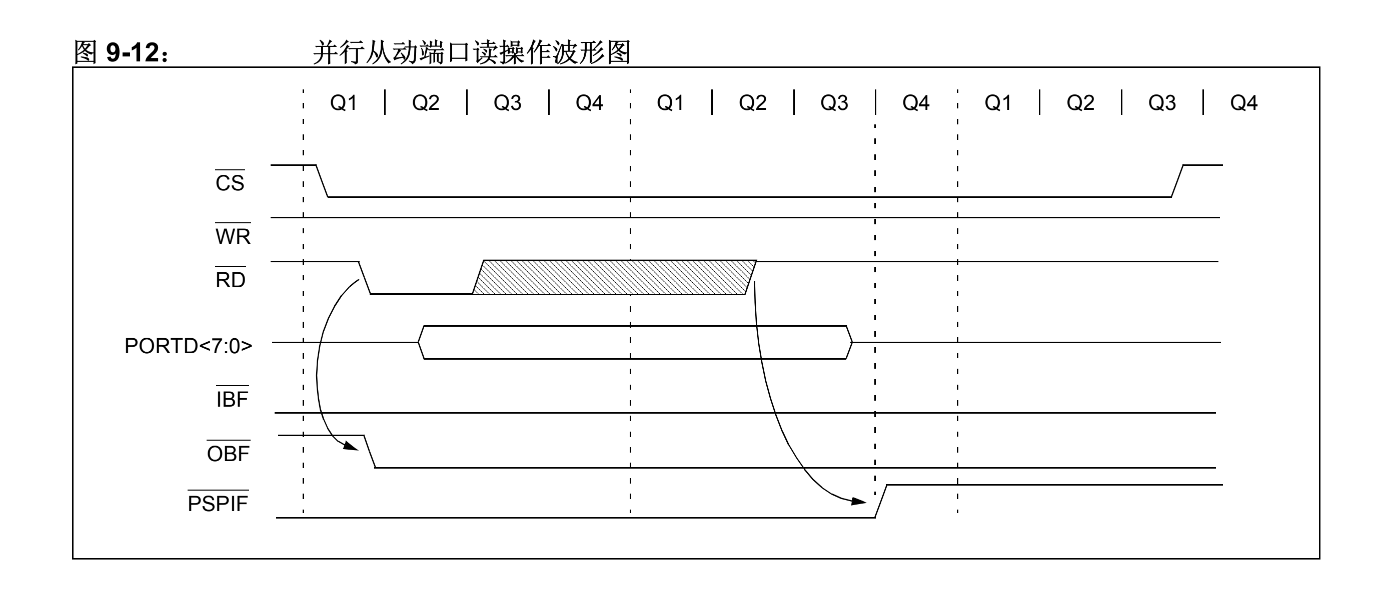


表 **9-11：** 与并行从动端口相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
PORTD	写操作时的端口数据锁存器；读操作时的端口引脚								xxxx xxxx	uuuu uuuu
LATD	LATD 数据输出位								xxxx xxxx	uuuu uuuu
TRISD	PORTD 数据方向位								1111 1111	1111 1111
PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -000	---- -000
LATE	—	—	—	—	—	LATE 数据输出位			---- -xxx	---- -uuu
TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE 数据方向位			0000 -111	0000 -111
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IF	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	00-- 0000	00-- 0000

图注： x = 未知，u = 不变， - = 未实现位，读为 0。阴影单元表示并行从动端口未使用。

注：

10.0 TIMER0 模块

Timer0 模块的特性如下：

- 可通过软件选择，作为 8 位或 16 位定时器 / 计数器
- 可读写
- 8 位软件可编程的专用预分频器
- 可选择内部或外部时钟源
- 8 位模式下 FFh 到 00h 的溢出中断， 16 位模式下 FFFFh 到 0000h 的溢出中断
- 外部时钟的边沿选择

图 10-1 是8位模式下Timer0模块的简化结构图，图 10-2 是 16 位模式下 Timer0 模块的简化结构图。

T0CON 寄存器（寄存器 10-1）是可读写寄存器，它可以控制所有Timer0的工作模式，包括预分频器的选择。

寄存器 10-1： T0CON：TIMER0 控制寄存器

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON**：Timer0 开 / 关控制位
1 = 使能 Timer0
0 = 禁止 Timer0

bit 6 **T08BIT**：Timer0 8 位 /16 位控制位
1 = Timer0 配置为 8 位定时器 / 计数器
0 = Timer0 配置为 16 位定时器 / 计数器

bit 5 **T0CS**：Timer0 时钟源选择位
1 = T0CKI 引脚上的输入信号作为时钟信号源
0 = 内部指令周期时钟 （CLKO）

bit 4 **T0SE**：Timer0 时钟源边沿选择位
1 = T0CKI 引脚下降沿传输时递增
0 = T0CKI 引脚上升沿传输时递增

bit 3 **PSA**：Timer0 预分频器分配位
1 = 未分配 TImer0 预分频器。 Timer0 时钟输入避开预分频器。
0 = 分配 TImer0 预分频器。 Timer0 时钟输入来自预分频器输出。

bit 2-0 **T0PS2:T0PS0**：Timer0 预分频比选择位
111 = 1:256 预分频值
110 = 1:128 预分频值
101 = 1:64 预分频值
100 = 1:32 预分频值
011 = 1:16 预分频值
010 = 1:8 预分频值
001 = 1:4 预分频值
000 = 1:2 预分频值

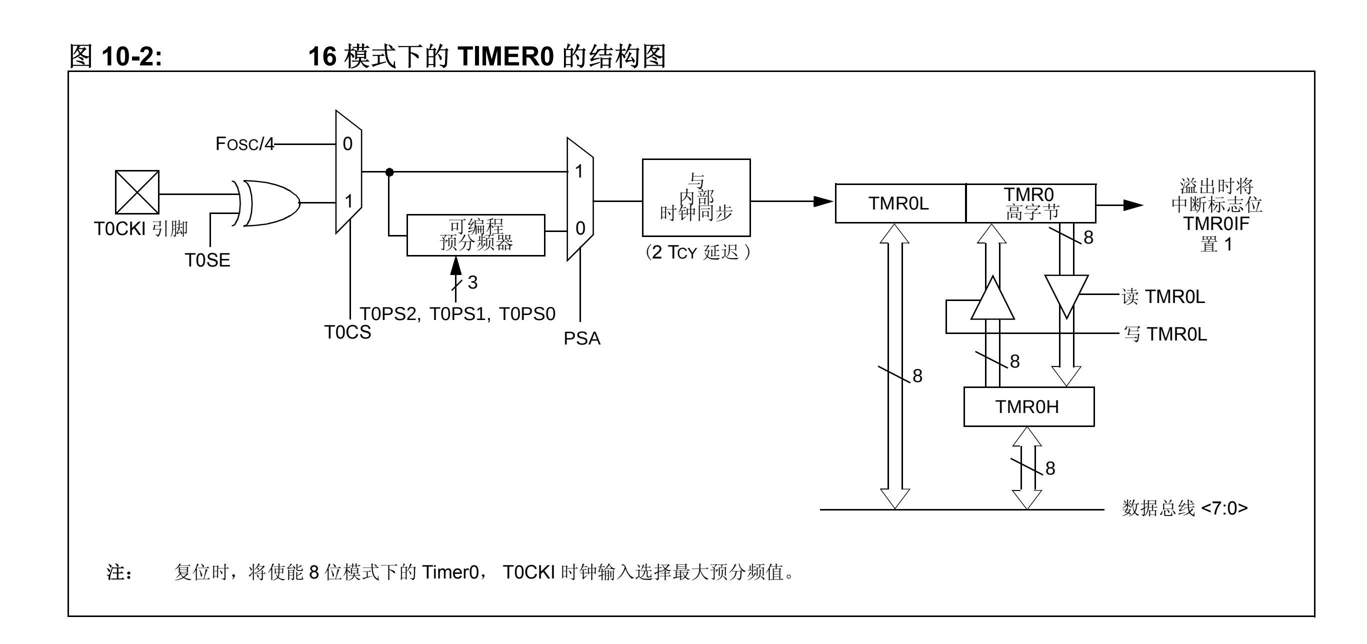
图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
- n = 上电复位时的值	1 = 置位	0= 清零	x = 未知位

图 10-1： 8 位模式下 TIMER0 的结构图

Fosc/4
0
1
RA4/T0CKI 引脚
T0SE
T0CS
可编程
预分频器
3
T0PS2, T0PS1, T0PS0
1
0
PSA
与内部
时钟
同步
(2 TCY 延迟）
数据总线
8
TMR0L
溢出时将中断
标志位 TMR0IF 置 1

注： 复位时，将使能 8 位模式下的 Timer0， T0CKI 时钟输入选择最大预分频值。

图 10-2: 16 模式下的 TIMER0 的结构图



注： 复位时，将使能 8 位模式下的 Timer0， T0CKI 时钟输入选择最大预分频值。

10.1 Timer0 工作模式

Timer0 既可用作定时器，亦可用作计数器。

将 T0CS 位清零即选择定时器模式。在定时器模式下，Timer0 模块在每个指令周期都会递增 （不使用预分频器）。如果写入 TMR0L 寄存器，在随后的两个指令周期内不再递增。用户可通过将校正值写入 TMR0L 寄存器来解决这个问题。

通过将 T0CS 位置 1，选择计数器模式。在计数器模式下， Timer0 可在 RA4/T0CKI 引脚的每个上升沿或下降沿的触发下递增。是上升沿触发还是下降沿触发由 Timer0 时钟源边沿选择位 （T0SE）决定。将 T0SE 位清零，即选择上升沿触发。下面讨论外部时钟输入的限制条件。

当 Timer0 使用外部时钟输入时，它必须满足一定的要求。这些要求确保外部时钟与内部相位时钟（Tosc）同步。在同步之后，Timer0 仍然要经过一个延时才会引发递增操作。

10.2 预分频器

Timer0 模块有一个 8 位计数器，用作预分频器。预分频器是不可读写的。

PSA 和 T0PS2:T0PS0 位决定预分频器的分配和预分频值。

PSA 位清零可将预分频器分配给 Timer0 模块。如果将预分频器分配给 Timer0 模块，可供选择的预分频值有 1:2、 1:4、……、 1:256。

当预分频器分配给 Timer0 模块时，所有对 TMR0L 寄存器进行写操作的指令（如 `CLRF TMR0`、`MOVWF TMR0`、`BSF TMR0, x` 等）都将使预分频器计数清零。

> 注： 当预分频器分配给 Timer0 时对 TMR0L 进行写操作将使预分频器计数清零，但不会改变预分频器的分配。

10.2.1 改变预分频器分配

预分频器的分配完全由软件控制（也就是说，在程序执行期间可以随时更改其分配）。

10.3 Timer0 中断

8 位模式下的 TMR0 寄存器发生 FFh 到 00h 溢出，或 16 位模式下的 TMR0 发生 FFFFh 到 0000h 的溢出时，将产生 TMR0 中断。该溢出将 TMR0IF 位置 1。可以通过 TMR0IE 位清零来屏蔽该中断。在重新使能此中断之前，必须用软件由 Timer0 模块的中断服务程序将 TMR0IE 位清零。由于定时器在休眠期间被关闭，所以 TMR0 中断无法把处理器从休眠状态唤醒。

10.4 16 位模式定时器读写

TMR0H并不是16位模式下定时器/计数器的高位字节，实际上是被缓存的 Timer0 高位字节 （见图 10-2）。不能直接读写 Timer0 计数器 / 定时器的高位字节。在读 TMR0L 时 Timer0 高位字节的内容将更新 TMR0H。由于读取高字节和低字节分两次连续进行，可能会有低字节向高字节的进位，需要验证读到的高字节和低字节的有效性，而现在可以一次读取 Timer0 的全部 16 位，就不需要验证读到的高字节和低字节的有效性了。

同样，可以使用 TMR0H 缓冲寄存器对 Timer0 的高位字节进行写入。使用 TMR0H 的内容更新 Timer0 的高位字节，同时写入TMR0L。这样一次就可以完成Timer0 的 16 位更新。

表 10-1： 与 TIMER0 相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
TMR0L	Timer0 模块的低位字节寄存器								xxxx xxxx	uuuu uuuu
TMR0H	Timer0 模块的高位字节寄存器								0000 0000	0000 0000
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	1111 1111
TRISA	—	PORTA 数据方向寄存器							-111 1111	-111 1111

图注： x = 未知， u = 不变， - = 未实现位，读为 0。阴影单元表示 Timer0 模块未使用。

注：

11.0 TIMER1 模块

Timer1 模块定时器 / 计数器具有如下特性：

- 16 位定时器 / 计数器
  （两个 8 位寄存器：TMR1H 和 TMR1L）
- 可读写 （TMR1H 和 TMR1L 寄存器均可）
- 可选择内部或外部时钟源
- FFFFh 到 0000h 的溢出中断
- CCP 模块特殊事件触发器复位

图 11-1 是 Timer1 模块的简化结构图。

寄存器 11-1 详细说明了 Timer1 控制寄存器。该寄存器控制 Timer1 模块的工作模式，它包含 Timer1 振荡器使能位 （T1OSCEN） 。可以通过置 1 或清零控制位 TMR1ON （T1CON<0>）来使能或者禁止 Timer1。

寄存器 11-1： T1CON：TIMER1 控制寄存器

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7 **RD16**：16 位读 / 写模式使能位
1 = 使能 Timer1 通过一次 16 位操作进行寄存器读 / 写功能
0 = 使能 Timer1 分两次 8 位操作进行寄存器读 / 写功能

bit 6 **未实现位**：读为 0

bit 5-4 **T1CKPS1:T1CKPS0**：Timer1 输入时钟预分频选择位
11 = 1:8 预分频值
10 = 1:4 预分频值
01 = 1:2 预分频值
00 = 1:1 预分频值

bit 3 **T1OSCEN**：Timer1 振荡器使能位
1 = Timer1 振荡器使能
0= Timer1 振荡器停振
关闭振荡器的反相器和反馈电阻以降低功耗。

bit 2 **$\overline{\text{T1SYNC}}$**：Timer1 外部时钟输入同步选择位
如果 TMR1CS = 1：
1 = 不同步外部时钟输入
0 = 同步外部时钟输入
如果 TMR1CS = 0：
此位被忽略。 TMR1CS = 0 时 Timer1 使用内部时钟。

bit 1 **TMR1CS**：Timer1 时钟源选择位
1 = 来自 RC0/T1OSO/T13CKI 引脚的外部时钟 （上升沿计数）
0 = 内部时钟 （FOSC/4）

bit 0 **TMR1ON**：Timer1 使能位
1 = 使能 Timer1
0 = 停止 Timer1

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
- n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

11.1 Timer1 工作模式

Timer1 有三种工作模式：

- 定时器模式
- 同步计数器模式
- 异步计数器模式

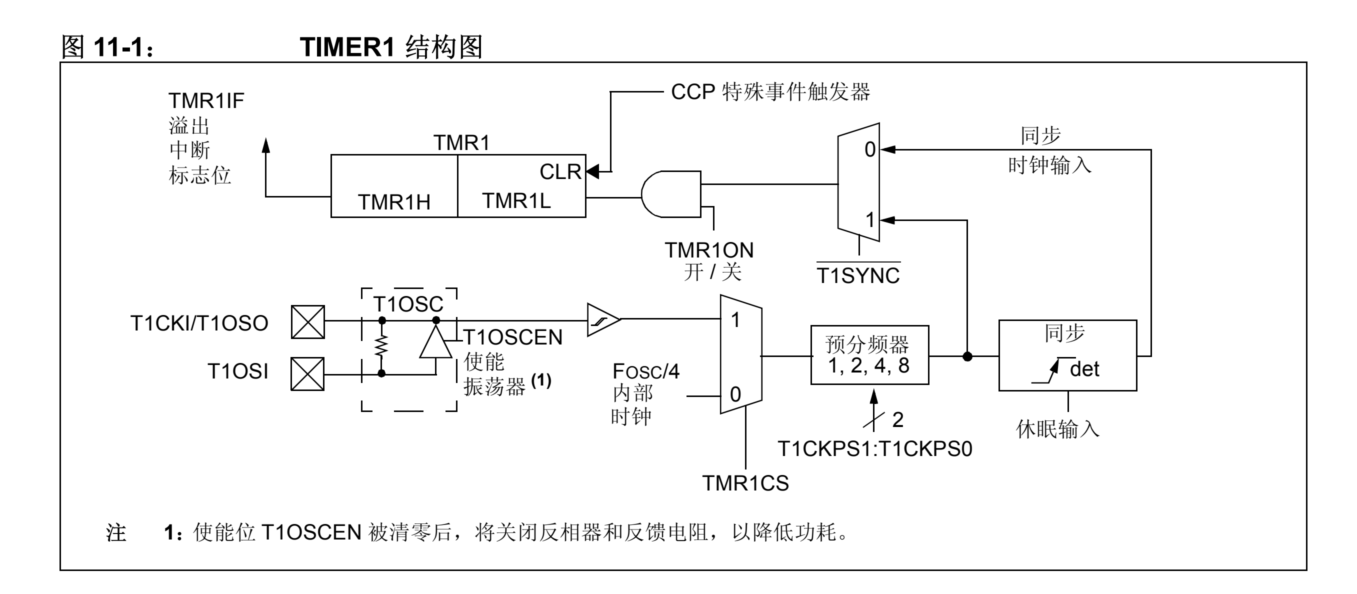
工作模式由时钟选择位 TMR1CS（T1CON<1>）决定。

如果 TMR1CS = 0，Timer1 在每个指令周期都会递增。如果 TMR1CS = 1，Timer1 在外部时钟输入或自带的 Timer1 振荡器（如果被使能）的每个上升沿的触发下递增。

如果 Timer1 振荡器被使能（T1OSCEN 置 1）的话，RC1/T1OSI 和 RC0/T1OSO/T1CKI 引脚就变成了输入引脚。即，忽略 TRISC<1:0> 的值，这两个引脚上都是 "0"。

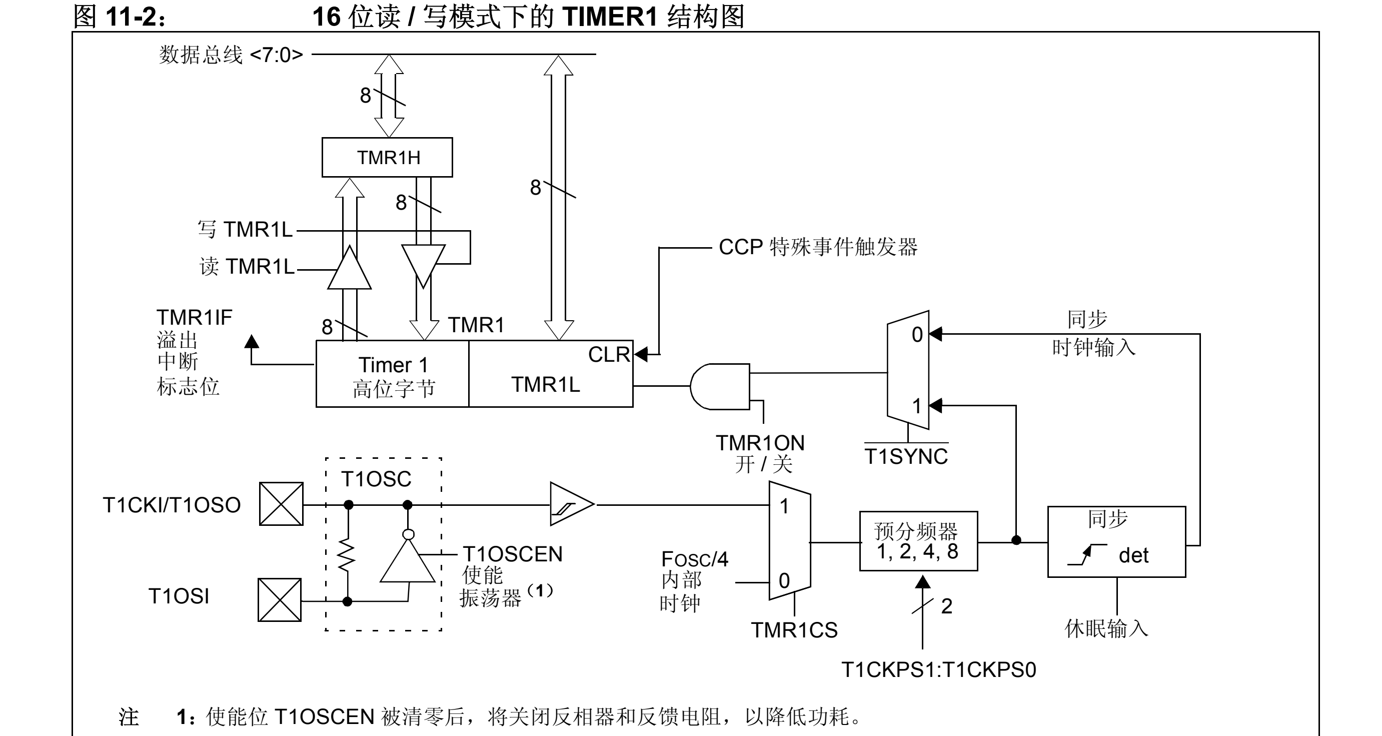
Timer1 也有内部 "复位输入"。这个复位输入可以由 CCP 模块产生（第 14.0 节）。

图 11-1： TIMER1 结构图



注 1：使能位 T1OSCEN 被清零后，将关闭反相器和反馈电阻，以降低功耗。

图 11-2： 16 位读 / 写模式下的 TIMER1 结构图



注 1：使能位 T1OSCEN 被清零后，将关闭反相器和反馈电阻，以降低功耗。

11.2 Timer1 振荡器

晶体振荡器电路跨接在 T1OSI 引脚（输入）和 T1OSO 引脚（放大器输出）之间。通过控制位 T1OSCEN（T1CON<3>）置 1 使能振荡器。该振荡器为低功耗振荡器，振荡频率最高达到 200 kHz。在休眠状态下可继续运行。一般建议使用 32 kHz 的石英晶体。表 11-1 给出 Timer1 振荡器的可选电容。

用户必须提供软件延时来确保 Timer1 振荡器的正常起振。

表 11-1：　　备用振荡器的电容选择

振荡类型	频率	C1	C2
LP	32 kHz	TBD[1]	TBD[1]
测试晶体：			
32.768 kHz	Epson C-001R32.768K-A		± 20 PPM

> 注 **1**：Microchip 建议在测试振荡器电路时从 33 pF 开始。
> **2**：大电容虽然可以提高振荡器的稳定性，但是同时会延长振荡器的起振时间。
> **3**：由于谐振器 / 晶体的特性各不相同，因此用户应向谐振器 / 晶体厂商咨询外围元件的正确参数。
> **4**：上述电容的值仅供设计参考。

11.3 Timer1 中断

TMR1 寄存器对（TMR1H:TMR1L）从 0000h 开始一直加到 FFFFh，然后转回 0000h 重新开始。如果使能了 Timer1 中断，则溢出将产生 TMR1 中断，并被锁存到中断标志位 TMR1IF （PIR1<0>）。可以通过对 TMR1 中断使能位 TMR1IE（PIE1<0>）置 1/ 清零来使能 / 禁止 Timer1 中断。

11.4 用 CCP 触发器输出复位 Timer1

如果 CCP 模块被设置为输出“特殊事件触发信号”（CCP1M3:CCP1M0 = `1011`）的比较模式，则该触发信号将复位 Timer1，并启动 A/D 转换（如果使能 A/D 模块的话）。

> 注：　CCP1模块的特殊事件触发信号不会将中断标志位 TMR1IF （PIR1<0>）置 1。

为了利用这一特性，Timer1 必须设置为定时器或同步计数器模式。如果 Timer1 在异步计数器模式下运行，则复位操作不起作用。

如果 Timer1 的写操作和 CCP1 模块的特殊事件触发复位操作同时发生，则写操作优先。

在这种工作模式下， CCPR1H:CCPR1L 这对寄存器实际上变成了 Timer1 的周期寄存器。

11.5 Timer1 16 位读 / 写模式

Timer1 可以针对 16 位读写进行配置（见图 11-2）。如果 RD16 控制位（T1CON<7>）被置 1， TMR1H 的地址被映射到 Timer1 高位字节的缓冲器。对 TMR1L 的读操作将把 Timer1 的高位字节内容装入 Timer1 高字节缓冲寄存器。这样，虽然两次读取之间发生进位可能会导致错误，但在这一模式下，由于可以准确读取整个 16 位 Timer1，因而不再需要验证低字节读取之后发生的高字节读取的有效性。

对 Timer1 的高位字节的写操作也必须通过 TMR1H 缓冲寄存器进行。使用 TMR1H 的内容更新 Timer1 高位字节，同时写入 TMR1L。这允许用户一次将 16 位内容写入 Timer1 的高位字节和低位字节。

在这一模式下不能直接读写 Timer1 的高位字节。所有读写都必须通过Timer1高位字节缓冲寄存器来进行。写入 TMR1H 不会清零 Timer1 预分频器。只有在写入 TMR1L 时才会将预分频器清零。

表 11-2： TIMER1 作为定时器 / 计数器时相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TMR1L	16 位 TMR1 寄存器低字节寄存器								xxxx xxxx	uuuu uuuu
TMR1H	16 位 TMR1 寄存器高字节寄存器								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu

图注： x = 未知，u = 不变， - = 未实现，读为 0。阴影单元表示 Timer1 模块未使用。

注 **1**：在 PIC18F2X 器件上保留了 PSPIF、 PSPIE 和 PSPIP 位；总是保持这些位为零。

12.0 TIMER2 模块

Timer2 模块定时器有以下特性：

- 8 位定时器 （TMR2 寄存器）
- 8 位周期寄存器 （PR2）
- 可读写 （TMR2 和 PR2 寄存器均可）
- 可软件编程的预分频器 （1:1、 1:4 和 1:16）
- 可软件编程的后分频器 （1:1 到 1:16）
- TMR2 与 PR2 匹配时中断
- SSP 模块可选用 TMR2 输出来产生时钟位移

Timer2 的控制寄存器，如寄存器 12-1 所示。通过清零 TMR2ON 控制位 （T2CON<2>）可以关闭 Timer2，以降低功耗。图 12-1 是 Timer2 模块的简化结构图。寄存器 12-1 是 Timer2 控制寄存器，该寄存器控制 Timer2 的预分频器和后分频器选择。

12.1 Timer2 工作模式

Timer2 可以与 CCP 模块配合使用，在 PWM 模式下用作 PWM 时基。TMR2 寄存器是可读写的，任何方式的单片机复位都会使之清零。输入时钟 （Fosc/4）有三种预分频选项，分别是 1:1、 1:4 或 1:16，可通过控制位 T2CKPS1:T2CKPS0 （T2CON<1:0>）来进行选择。TMR2 的匹配输出通过 4 位后分频器（从 1:1 到 1:16 的后分频值）生成 TMR2 中断 （锁存在 TMR2IF 标志位 （PIR1<1>））。

当发生下列任何一种情况时，都会对预分频器和后分频器计数器同时清零：

- 对 TMR2 寄存器进行写操作
- 对 T2CON 寄存器进行写操作
- 任何方式的单片机复位 （上电复位、 $\overline{\text{MCLR}}$ 复位、看门狗定时器复位或者欠压复位）

对 T2CON 进行写操作时不会使 TMR2 清零。

寄存器 12-1： T2CON：TIMER2 控制寄存器

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7 **未实现位**：读为 0

bit 6-3 **TOUTPS3:TOUTPS0**：Timer2 输出后分频选择位
`0000` = 1:1 后分频值
`0001` = 1:2 后分频值
•
•
•
`1111` = 1:16 后分频值

bit 2 **TMR2ON**：Timer2 使能位
`1` = 使能 Timer2
`0` = 停止 Timer2

bit 1-0 **T2CKPS1:T2CKPS0**：Timer2 时钟预分频选择位
`00` = 预分频值为 1
`01` = 预分频值为 4
`1x` = 预分频值为 16

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
- n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

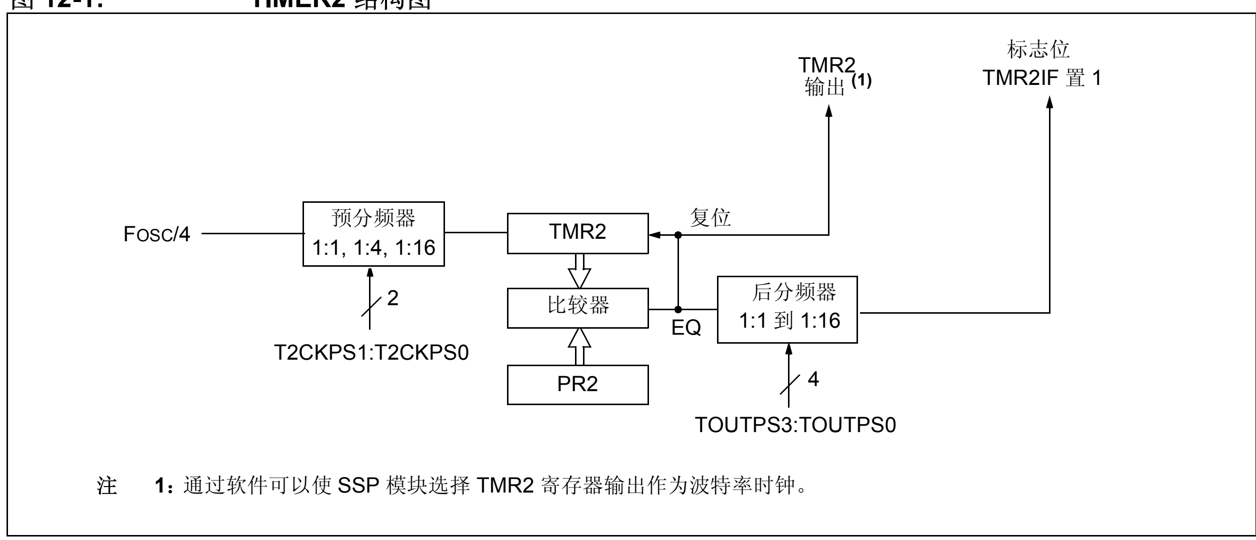
12.2 Timer2 中断

Timer2 模块有一个 8 位周期寄存器 PR2。Timer2 从 00h 开始递增，直到与 PR2 匹配为止，然后在下一计数周期开始时复位为 00h。 PR2 为可读写寄存器。复位会使 PR2 寄存器初始化为 FFh。

12.3 TMR2 输出

在后分频器之前，将 TMR2 输出馈送至同步串行口模块，亦可将其用于生成位移时钟。

图 12-1: TIMER2 结构图



注 1：通过软件可以使 SSP 模块选择 TMR2 寄存器输出作为波特率时钟。

表 12-1： TIMER2 作为定时器 / 计数器时相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TMR2	Timer2 模块的寄存器								0000 0000	0000 0000
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
PR2	Timer2 周期寄存器								1111 1111	1111 1111

图注： x = 未知， u = 不变， - = 未实现读为 0。阴影单元表示 Timer2 模块未使用。

注 1：在 PIC18F2X 器件上保留了 PSPIF、 PSPIE 和 PSPIP 位；这些位始终保持为零。

13.0 TIMER3 模块

Timer3 模块定时器 / 计数器有以下特性：

- 16 位定时器 / 计数器
（两个 8 位寄存器：TMR3H 和 TMR3L）
- 可读写 （TMR3H 和 TMR3L 寄存器均可）
- 内部或外部时钟选择
- FFFFh 到 0000h 的溢出中断
- CCP 模块触发器复位

图 13-1 是 Timer3 模块的简化结构图。

寄存器 13-1 中是 Timer3 控制寄存器。该寄存器控制 Timer3 模块的工作模式，并设置 CCP 时钟源。

寄存器 11-1 是 Timer1 控制寄存器。该寄存器控制 Timer1 模块的工作模式，它包含 Timer1 振荡器使能位（T1OSCEN）， Timer1 振荡器的输出可以用作 Timer3 的时钟源。

寄存器 13-1： T3CON：TIMER3 控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

bit 7 **RD16**：16 位读 / 写模式使能位
1 = 使能 Timer3 通过一次 16 位操作进行寄存器读 / 写
0 = 使能 Timer3 分两次 8 位操作进行寄存器读 / 写

bit 6-3 **T3CCP2:T3CCP1**：Timer3 和 Timer1 到 CCPx 的使能位
1x = Timer3 是比较 / 捕捉 CCP 模块的时钟源
01 = Timer3 是比较 / 捕捉 CCP2 模块的时钟源，
Timer1 是比较 / 捕捉 CCP1 模块的时钟源
00 = Timer1 是比较 / 捕捉 CCP 模块的时钟源

bit 5-4 **T3CKPS1:T3CKPS0**：Timer3 输入时钟预分频选择位
11 = 1:8 预分频值
10 = 1:4 预分频值
01 = 1:2 预分频值
00 = 1:1 预分频值

bit 2 **$\overline{\text{T3SYNC}}$**：Timer3 外部时钟输入同步控制位
（不适用于系统时钟来自 Timer1/Timer3 的场合）
如果 TMR3CS = 1：
1 = 不同步外部时钟输入
0 = 同步外部时钟输入
如果 TMR3CS = 0：
此位被忽略。 TMR3CS = 0 时 Timer3 使用内部时钟。

bit 1 **TMR3CS**：Timer3 时钟源选择位
1 = 使用 Timer1 振荡器输出或 T1CKI 作为外部时钟输入
（第一个下降沿后的上升沿计数）
0 = 内部时钟 （Fosc/4）

bit 0 **TMR3ON**：Timer3 使能位
1 = 使能 Timer3
0 = 停止 Timer3

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为 0	
- n = 上电复位时的值	1 = 置位	0 = 清零	x = 未知

13.1 Timer3 工作模式

Timer3 有三种工作模式：

- 定时器模式
- 同步计数器模式
- 异步计数器模式

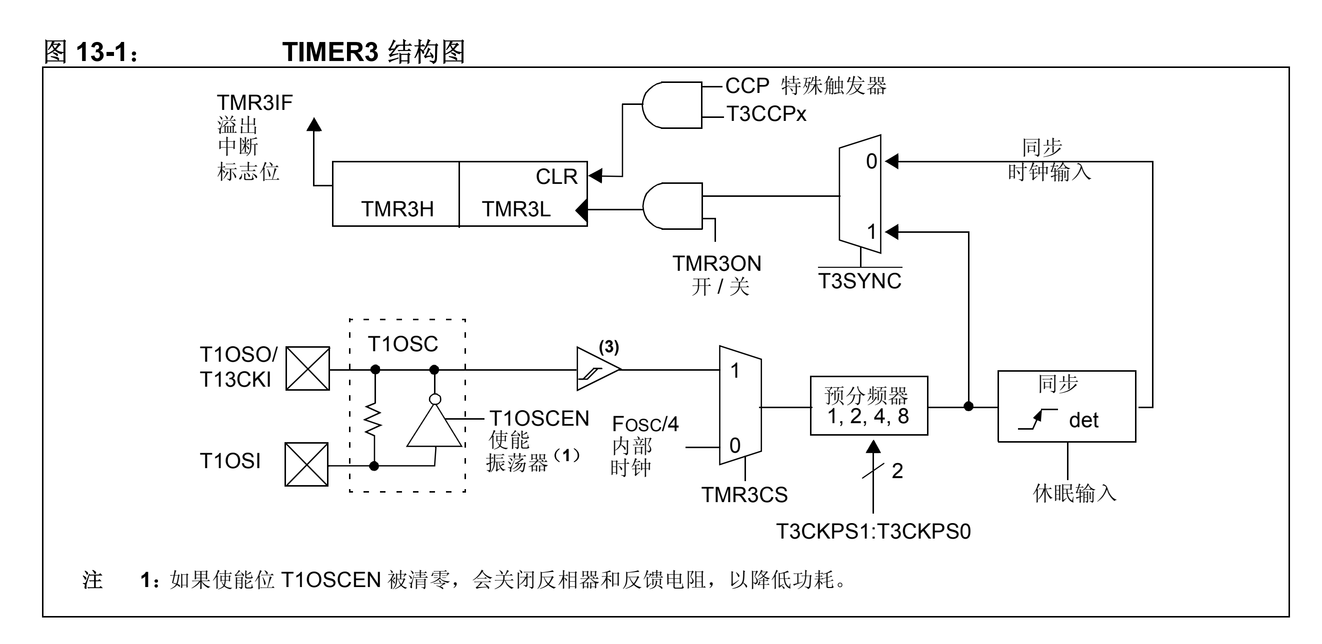
工作模式由时钟选择位 TMR3CS（T3CON<1>）确定。

如果 TMR3CS = 0，Timer3 在每个指令周期都会递增。如果 TMR3CS = 1， Timer3 会在 Timer1 外部时钟输入或 Timer1 振荡器 （如果使能的话）的每个上升沿的触发下递增。

如果使能 Timer1 振荡器 （T1OSCEN 置 1）， RC1/T1OSI 和 RC0/T1OSO/T1CKI 引脚成为输入引脚。即，忽略 TRISC<1:0> 的值，这两个引脚读为 0。

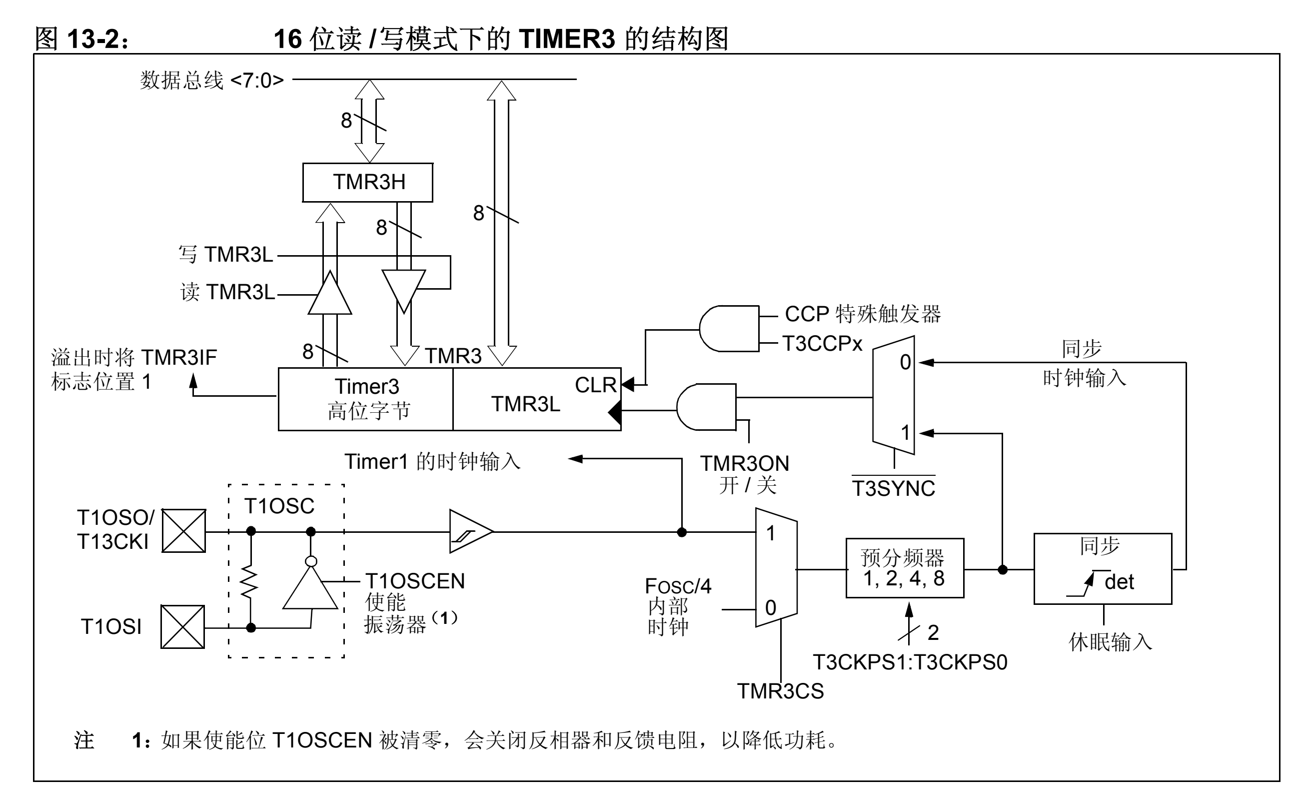
Timer3 也有内部 “复位输入”。该复位信号可由 CCP 模块 （第 14.0 节）产生。

图 13-1： **TIMER3 结构图**



注 1：如果使能位 T1OSCEN 被清零，会关闭反相器和反馈电阻，以降低功耗。

图 13-2： **16 位读 / 写模式下的 TIMER3 的结构图**



注 1：如果使能位 T1OSCEN 被清零，会关闭反相器和反馈电阻，以降低功耗。

13.2 Timer1 振荡器

Timer1 振荡器可用作 Timer3 的时钟源。通过将 T1OSCEN 位（T1CON<3>）置 1，可使能 Timer1 振荡器。该振荡器是一个低功耗振荡器，频率最高可达 200 kHz。详细内容请参见第 11.0 节 。

13.3 Timer3 中断

TMR3 寄存器对（TMR3H:TMR3L）从 0000h 开始计数，到 FFFFh 为止，然后从 0000h 重新开始计数。如果 TMR3 中断被使能，则溢出时会产生中断，并被锁存在中断标志位 TMR3IF（PIR2<1>）。通过置 1/ 清零 TMR3 中断使能位 TMR3IE（PIE2<1>）可以使能 / 禁止 TMR3 中断。

13.4 用 CCP 触发器输出复位 Timer3

如果 CCP 模块被设置为输出"特殊事件触发信号"（CCP1M3:CCP1M0 = 1011）的比较模式，则该触发信号会复位 Timer3。

注：	CCP 模块的特殊事件触发信号不会将中断标志位 TMR3IF（PIR1<0>）置 1。

为了利用这一特性，Timer3 必须设置为定时器或同步计数器模式。如果 Timer3 在异步计数器模式下运行，复位不会起作用。当 Timer3 的写操作与 CCP1 的特殊事件触发复位操作同时发生时，写操作优先。在这种模式下，CCPR1H:CCPR1L 寄存器对实际上变成了 Timer3 的周期寄存器。

表 13-1： TIMER3 作为定时器 / 计数器时相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	---0 0000
PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	---0 0000
IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	---1 1111
TMR3L	16 位 TMR3 寄存器低 8 位的保持寄存器								xxxx xxxx	uuuu uuuu
TMR3H	16 位 TMR3 寄存器高 8 位的保持寄存器								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0000 0000	uuuu uuuu

图注： x = 未知，u = 不变， - = 未实现位，读为 0。阴影单元表示 Timer1 模块未使用。

注：

14.0 比较 / 捕捉 /PWM （CCP）模块

每个 CCP （捕捉 / 比较 /PWM）模块有一个 16 位寄存器，它可以用作 16 位捕捉寄存器、16 位比较寄存器或 PWM 主 / 从占空比寄存器。表 14-1 显示了各 CCP 模块工作模式的定时器资源。

除了特殊事件触发器之外，CCP1的操作和CCP2相同。因此，以下各节中描述的 CCP 模块操作是针对 CCP1 而言的。

表 14-2 显示了 CCP 模块之间的相互关系。

寄存器 14-1： CCP1CON 寄存器 /CCP2CON 寄存器

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DCxB1	DCxB0	CCPxM3	CCPxM2	CCPxM1	CCPxM0
bit 7							bit 0

bit 7-6 **未实现**：读作 0

bit 5-4 **DCxB1:DCxB0**：PWM 占空比 bit 1 和 bit 0
捕捉模式：
未用
比较模式：
未用
脉宽调制模式：
它们是 10 位 PWM 占空比的两个 LSb （bit 1 和 bit 0）。占空比的高 8 位 （DCx9:DCx2）在 CCPRxL 中。

bit 3-0 **CCPxM3:CCPxM0**：CCPx 模式选择位
0000= 禁用捕捉 / 比较 /PWM （即复位 CCPx 模块）
0001= 保留
0010= 比较模式，匹配时翻转输出 （CCPxIF 置 1）
0011= 保留
0100= 捕捉模式，每个下降沿发生
0101= 捕捉模式，每个上升沿发生
0110= 捕捉模式，每 4 个上升沿发生
0111= 捕捉模式，每 16 个上升沿发生
1000= 比较模式，
CCP 引脚初始为低电平，比较相符时，强制 CCP 引脚输出高电平 （CCPIF 位置 1）
1001= 比较模式，
CCP 引脚初始为高电平，比较相符时，强制 CCP 引脚输出低电平 （CCPIF 位置 1）
1010= 比较模式，
比较相符时，产生软件中断 （CCPIF 位置 1， CCP 引脚不受影响）
1011= 比较模式，
特殊事件触发 （CCPIF 位置 1）
11xx= PWM 模式

图注：			
R= 可读位	W= 可写位	U= 未实现位，读作 0	
-n= 上电复位时的值	1= 置位	0= 清零	x= 未知

14.1 CCP1 模块

捕捉 / 比较 /PWM 寄存器 1（CCPR1）由两个 8 位寄存器组成：CCPR1L （低字节）和 CCPR1H （高字节）。CCP1CON 寄存器控制 CCP1 的操作。所有位均可读写。

14.2 CCP2 模块

捕捉 / 比较 /PWM 寄存器 2（CCPR2）由两个 8 位寄存器组成：CCPR2L （低字节）和 CCPR2H （高字节）。CCP2CON 寄存器控制 CCP2 的操作。所有位均可读写。

表 14-1： CCP 模式－定时器资源

CCP 模式	定时器资源
捕捉	Timer1 或 Timer3
比较	Timer1 或 Timer3
PWM	Timer2

表 14-2： 两个 CCP 模块的相互关系

CCPx 模式	CCPy 模式	相互关系
捕捉	捕捉	TMR1 或 TMR3 时基。每个 CCP 的时基可以各不相同。
捕捉	比较	可为特殊事件触发器配置比较模式， 用来对 TMR1 或 TMR3 清零 （取决于使用哪一个时基）。
比较	比较	可为特殊事件触发器配置比较模式， 用来对 TMR1 或 TMR3 清零 （取决于使用哪一个时基）。
PWM	PWM	PWM 将具有相同的频率和更新速率 （TMR2 中断）。
PWM	捕捉	无
PWM	比较	无

14.3 捕捉模式

在捕捉模式下，当 RC2/CCP1 引脚上有事件发生时，CCPR1H:CCPR1L 即捕捉 TMR1 或 TMR3 寄存器的 16 位计数值。事件定义如下：

- 每个下降沿发生
- 每个上升沿发生
- 每 4 个上升沿发生
- 每 16 个上升沿发生

由控制位 CCP1M3:CCP1M0 （CCP1CON<3:0>）来选择上述 4 种事件之一。当发生捕捉事件时，中断请求标志位 CCP1IF（PIR1<2>）置 1 ；该位必须用软件清零。如果在读出寄存器CCPR1中的值之前发生另一个捕捉，那么之前捕捉的值将会被覆盖。

14.3.1 CCP 引脚配置

捕捉模式下，应通过将 TRISC<2> 位置 1 将 RC2/CCP1 引脚设置为输入引脚。

注：	如果 RC2/CCP1 引脚被设置为输出引脚，则对该端口的写操作可能引发一个捕捉事件。

14.3.2 TIMER1/TIMER3 模式选择

用于捕捉功能的定时器 （Timer1 和 / 或 Timer3）必须运行在定时器模式或同步计数器模式下。在异步计数器模式下，可能无法进行捕捉操作。用于每个 CCP 模块的定时器在 T3CON 寄存器中进行选择。

14.3.3 软件中断

捕捉模式的改变会产生错误的捕捉中断。用户应保持 CCP1IE 位 （PIE1<2>）为 0 以免发生错误中断，并且应该在任何操作模式改变后清零标志位 CCP1IF。

14.3.4 CCP 预分频器

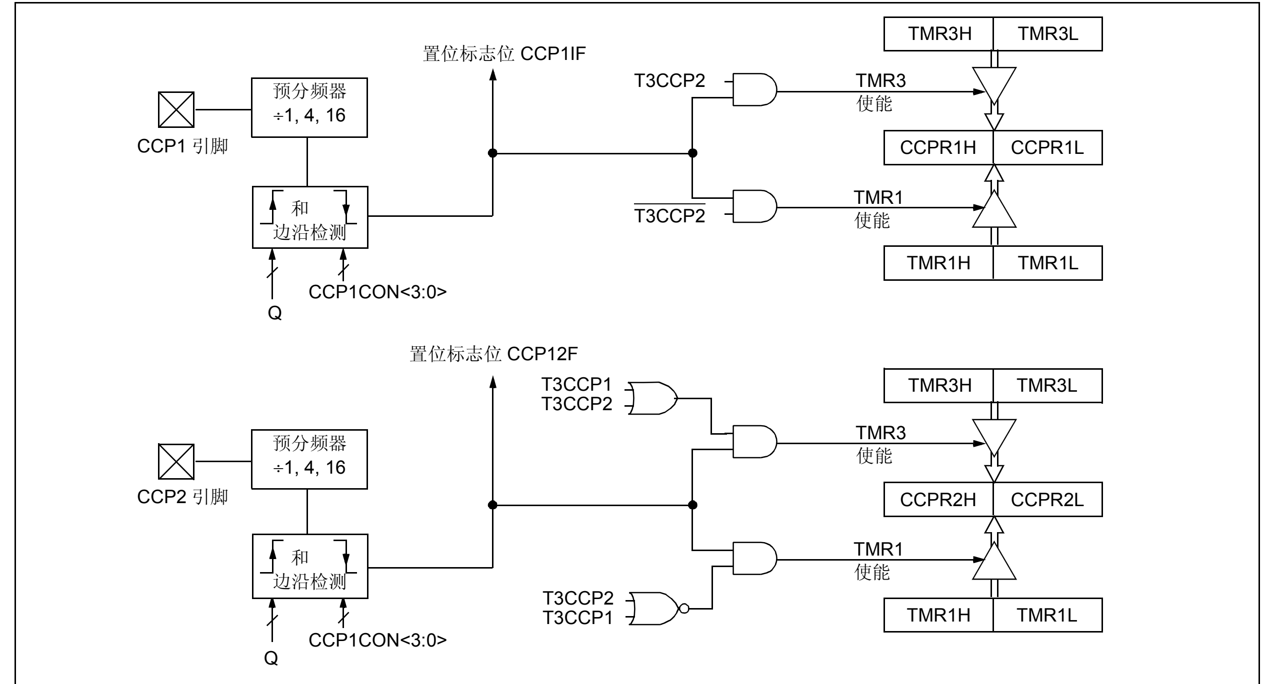
通过设置 CCP1M3:CCP1M0 位可以选择四种预分频值设置。只要 CCP 模块关闭或没有设置为捕捉模式，就可将预分频器的计数器清零。这就意味着任何复位都可以将预分频器计数器清零。

从一个捕捉预分频器切换到另一个捕捉预分频器可能产生中断。而且，预分频器计数器不会被清零，因此第一次捕捉可能是来自一个非零的预分频器。例 14-1 是切换捕捉预分频器时建议采用的方法。该例也对预分频器的计数器清零，这样就不会产生 “错误的”中断。

例 14-1： 捕捉预分频器的转换

```
CLRF    CCP1CON, F ; Turn CCP module off
MOVLW  NEW_CAPT_PS ; Load WREG with the
                   ; new prescaler mode
                   ; value and CCP ON
MOVWF  CCP1CON     ; Load CCP1CON with
                   ; this value
```

图 14-1： 捕捉模式工作结构图



14.4 比较模式

比较模式下， 16 位 CCPR1 （CCPR2）寄存器的值随时与 TMR1 或 TMR3 寄存器对的值相比较。当两者相符时， RC2/CCP1 （RC1/CCP2）引脚将：

- 变为高电平
- 变为低电平
- 翻转输出 （高电平变为低电平或低电平变为高电平）
- 保持不变

引脚的状态取决于控制位 CCP1M3:CCP1M0 （CCP2M3:CCP2M0）的值。同时，中断标志位CCP1IF （CCP2IF）置 1。

14.4.1 CCP 引脚配置

用户必须通过将相应的 TRISC 位清零，将 CCPx 引脚配置为输出引脚。

注：	清零 CCP1CON 寄存器将强制 RC2/CCP1 比较输出锁存器为默认低电平。这不是 PORTC I/O 数据锁存器。

14.4.2 TIMER1/TIMER3 模式选择

如果 CCP 模块使用比较功能，Timer1 和 / 或 Timer3 必须工作在定时器模式或同步计数器模式。在异步计数器模式下，可能无法进行比较操作。

14.4.3 软件中断模式

当选择了产生软件中断时， CCP1 引脚上的电平不受影响。 CCP 中断使能时，只会产生一个 CCP 中断。

14.4.4 特殊事件触发器

在这一模式下，将产生一个内部硬件触发信号，可用来触发一个操作。

CCP1 的特殊事件触发器输出使 TMR1 寄存器对复位。这将使 CCPR1 寄存器有效地成为 Timer1 的 16 位可编程周期寄存器。

CCPx 的特殊事件触发器输出使 TMR1 或 TMR3 寄存器对复位。另外，如果 A/D 模块使能，则 CCP2 特殊事件触发器将启动 A/D 转换。

注：	CCP2 模块的特殊事件触发器不会将 Timer1 或 Timer3 的中断标志位置 1。

图 **14-2：** 比较模式工作结构图

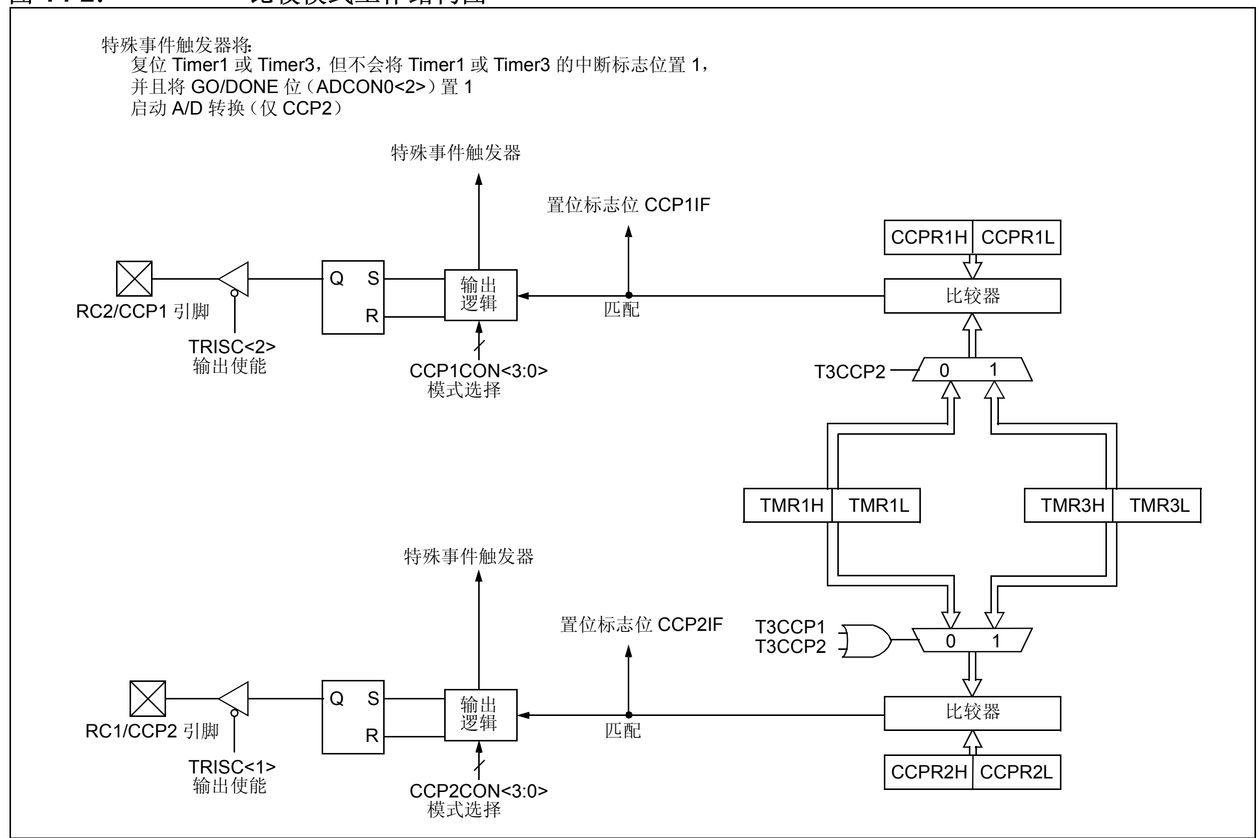


表 **14-3**： 与捕捉、比较、 **TIMER1** 和 **TIMER3** 相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	所有其他复位的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF[1]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE[1]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP[1]	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TRISC	PORTC 数据方向寄存器								1111 1111	1111 1111
TMR1L	16 位 TMR1 寄存器低位字节的保持寄存器								xxxx xxxx	uuuu uuuu
TMR1H	16 位 TMR1 寄存器高位字节的保持寄存器								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu
CCPR1L	比较 / 捕捉 /PWM 寄存器 1 （LSB）								xxxx xxxx	uuuu uuuu
CCPR1H	比较 / 捕捉 /PWM 寄存器 1 （MSB）								xxxx xxxx	uuuu uuuu
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
CCPR2L	比较 / 捕捉 /PWM 寄存器 2 （LSB）								xxxx xxxx	uuuu uuuu
CCPR2H	比较 / 捕捉 /PWM 寄存器 2 （MSB）								xxxx xxxx	uuuu uuuu
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000
PIR2	—	—	—	EEIE	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	---0 0000
PIE2	—	—	—	EEIF	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	---0 0000
IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	---1 1111
TMR3L	16 位 TMR3 寄存器低位字节的保持寄存器								xxxx xxxx	uuuu uuuu
TMR3H	16 位 TMR3 寄存器高位字节的保持寄存器								xxxx xxxx	uuuu uuuu
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0000 0000	uuuu uuuu

图注： x= 未知， u= 不变， - = 未实现，读作 0。阴影单元不适用于捕捉和 Timer1。

注 1：PIC18F2x2 器件保留 PSPIF、 PSPIE 和 PSPIP 位；始终将这些位清零。

14.5 PWM 模式

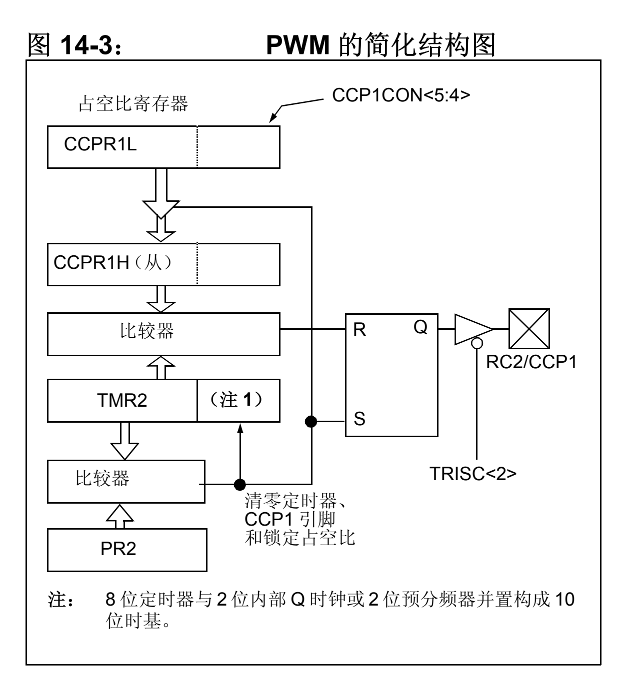
在脉冲宽度调制（Pulse Width Modulation，PWM）模式下， CCP1 引脚可产生分辨率高达 10 位的 PWM 输出。因为 CCP1 引脚与 PORTC 数据锁存器复用，所以 TRISC<2>位必须清零以使CCP1引脚为输出状态。

注：	清零 CCP1CON 寄存器将强制 CCP1 PWM 输出锁存器为默认低电平。这不是 PORTC I/O 数据锁存器。

图 14-3 是 PWM 模式下工作的 CCP 模块简化结构图。

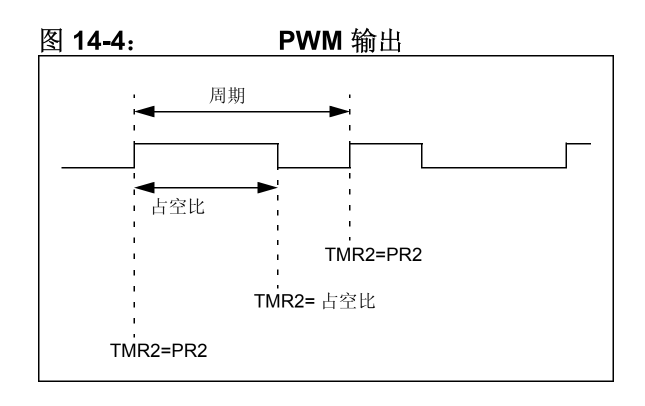
有关将 CCP 模块设置成 PWM 操作的详细步骤，请参见第 14.5.3 节。

图 14-3： PWM 的简化结构图



一个 PWM 输出 （图 14-4）包含一个时基 （周期）和一段输出高电平的时间 （占空比）。 PWM 的频率是周期的倒数。

图 14-4： PWM 输出



14.5.1 PWM 周期

PWM 周期可通过写入 PR2 寄存器来指定。可用以下公式计算 PWM 周期：

PWM 周期 = (PR2) + 1] • 4 • TOSC • (TMR2 预分频值)

PWM 频率定义为 1/[PWM 周期]。

当 TMR2 等于 PR2 时，在下一递增计数周期中将产生下面三个事件：

- 清零 TMR2
- 将 CCP1 引脚置 1 （例外情况：如果 PWM 占空比 = 0%， CCP1 引脚不被置 1）
- PWM 占空比从 CCPR1L 被锁定为 CCPR1H

注：	确定 PWM 频率时不使用 Timer2 后分频器（参阅第 12.0 节）。使用后分频器时，其伺服更新速率可与 PWM 输出频率不同。

14.5.2 PWM 占空比

PWM 占空比可通过向 CCPR1L 寄存器和 CCP1CON<5:4> 位写入来指定。最高分辨率可达 10 位。CCPR1L 包含 8 位 MSb，CCP1CON<5:4> 包含 2 位 LSb。这 10 位值由 CCPR1L:CCP1CON<5:4> 来表征。计算 PWM 占空比的公式如下：

PWM 占空比 = (CCPR1L:CCP1CON<5:4>) • TOSC • (TMR2 预分频值)

可以在任何时候写入 CCPR1L 和 CCP1CON<5:4>，但直到 PR2 与 TMR2 中的值相符 （例如，当周期结束）时，占空比的值才被锁存到 CCPR1H。在 PWM 模式下， CCPR1H 是只读寄存器。

CCPR1H 寄存器和一个 2 位的内部锁存器用于为 PWM 占空比提供双重缓冲。对于 PWM 的无毛刺操作，双重缓冲是很必要的。

当 CCPR1H 和 2 位锁存器的值与附加了内部 2 位 Q 时钟或 2 位 TMR2 预分频器的 TMR2 相符时，CCP1 引脚被清零。

对于给定的 PWM 频率，其最大分辨率 （位）为：

$$\text{PWM 分辨率（最大）} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{位}$$

注：	如果 PWM 占空比的值大于 PWM 周期，不会将 CCP1 引脚清零。

14.5.3 设置 PWM 操作

通过以下步骤将 CCP 模块配置为 PWM 操作：

1. 写入 PR2 寄存器以设定 PWM 周期。
2. 写入 CCPR1L 寄存器和 CCP1CON<5:4> 位以设置 PWM 占空比。
3. 将 TRISC<2> 位清零以将 CCP1 引脚设为输出。
4. 写入 T2CON 以设置 TMR2 预分频值并使能 Timer2。
5. 将 CCP1 模块配置为 PWM 模式。

表 14-4： 40 MHz 下的 PWM 频率和分辨率示例

PWM 频率	2.44 kHz	9.77 kHz	39.06 kHz	156.25 kHz	312.50 kHz	416.67 kHz
定时器预分频值 （1, 4, 16）	16	4	1	1	1	1
PR2 值	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
最大分辨率 （位）	14	12	10	8	7	6.58

表 14-5： 与 PWM 和 TIMER2 相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	所有其他复位的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TRISC	PORTC 数据方向寄存器								1111 1111	1111 1111
TMR2	Timer2 模块寄存器								0000 0000	0000 0000
PR2	Timer2 模块周期寄存器								1111 1111	1111 1111
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
CCPR1L	比较 / 捕捉 /PWM 寄存器 1 （LSB）								xxxx xxxx	uuuu uuuu
CCPR1H	比较 / 捕捉 /PWM 寄存器 1 （MSB）								xxxx xxxx	uuuu uuuu
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
CCPR2L	比较 / 捕捉 /PWM 寄存器 2 （LSB）								xxxx xxxx	uuuu uuuu
CCPR2H	比较 / 捕捉 /PWM 寄存器 2 （MSB）								xxxx xxxx	uuuu uuuu
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

图注： x= 未知， u= 不变， -= 未实现，读作 0。阴影单元不适用于 PWM 和 Timer2。

注 1：PIC18F2x2 器件保留 PSPIF、 PSPIE 和 PSPIP 位；始终将这些位清零。

注：

15.0 主同步串行口 （MSSP）模块

15.1 主 SSP （MSSP）模块概述

主同步串行口 （MSSP）模块是用于同其他外设或单片机器件进行通讯的串行接口。这些外设包括串行 EEPROM、移位寄存器、显示驱动器和 A/D 转换器等。MSSP 模块有两种工作模式：

- 串行外设接口 （SPI）
- 内部互联 （I^2C）
  - 全主控模式
  - 从动模式 （全局地址呼叫）

I^2C 接口通过硬件支持下列模式：

- 主控模式
- 多主控模式
- 从动模式

15.2 控制寄存器

MSSP 模块有三个相关的寄存器。包括一个状态寄存器（SSPSTAT）和两个控制寄存器（SSPCON1 和 SSPCON2）。根据 MSSP 模块是在 SPI 模式还是 I^2C 模式下工作，这些寄存器的用途及其配置位将大相径庭。

下面各节会提供其他细节。

15.3 SPI 模式

SPI 模式允许同时同步发送和接收 8 位数据。支持 SPI 的所有四种模式。一般用以下三个引脚来实现通讯：

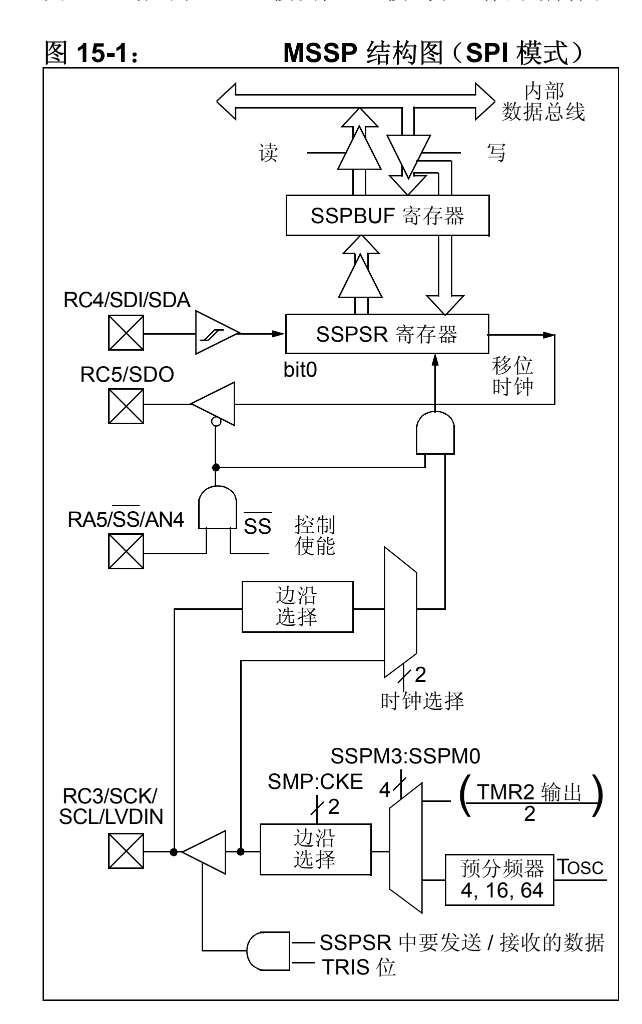
- 串行数据输出 （Serial Data Out， SDO）－ RC5/SDO
- 串行数据输入 （Serial Data In， SDI）－ RC4/SDI/SDA
- 串行时钟 （Serial Clock， SCK）－ RC3/SCK/SCL/LVDIN

此外，在从动模式下工作可能会用到第四个引脚：

- 从动选择 （$\overline{SS}$）－ RA5/$\overline{SS}$/AN4

图 15-1给出了MSSP模块在SPI模式下工作的结构图。

图 15-1： MSSP 结构图（SPI 模式）



15.3.1 寄存器

MSSP 模块有四个寄存器用于 SPI 模式。它们是：

- MSSP 控制寄存器 1 （SSPCON1）
- MSSP 状态寄存器 （SSPSTAT）
- 串行接收 / 发送缓冲器 （SSPBUF）
- MSSP 移位寄存器 （SSPSR）－ 不可直接存取

SSPCON1 和 SSPSTAT 是在 SPI 模式下工作的控制寄存器和状态寄存器。 SSPCON1 寄存器是可读写的。SSPSTAT 的低六位是只读的。SSPSTAT 的高两位是可读写的。

SSPSR 是用来将数据移入或移出的移位寄存器。SSPBUF 是缓冲寄存器，可用于读写数据字节。

接收时，SSPSR 和 SSPBUF 共同构成一个双重缓冲接收器。当 SSPSR 接收完整的字节后，将其送入 SSPBUF，同时将中断标志位 SSPIF 置 1。

在发送中，SSPBUF 并不是双重缓冲的。写入 SSPBUF 的同时将写入 SSPBUF 和 SSPSR。

寄存器 15-1： SPI 模式下的 MSSP 状态寄存器：SSPSTAT

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

bit 7 **SMP：**采样位
SPI 主控模式：
1= 在数据输出期间的结束时采样输入数据
0= 在数据输出期间的中段采样输入数据
SPI 从动模式：
SPI 在从动模式下， SMP 必须被清零

bit 6 **CKE：**SPI 时钟沿选择
当 CKP=0 时：
1= 在 SCK 上升沿发送数据
0= 在 SCK 下降沿发送数据
当 CKP=1 时：
1= 在 SCK 下降沿发送数据
0= 在 SCK 上升沿发送数据

bit 5 **D/$\overline{A}$：**数据 / 地址位
只在 I^2C 模式中使用

bit 4 **P：**停止位
只在 I^2C 模式中使用。当 MSSP 模块被禁止 （SSPEN 清零）时该位被清零。

bit 3 **S：**启动位
只在 I^2C 模式中使用

bit 2 **R/$\overline{W}$：**读 / 写位信息
只在 I^2C 模式中使用

bit 1 **UA：**更新地址
只在 I^2C 模式中使用

bit 0 **BF：**缓冲器满状态位 （仅在接收模式下）
1= 表示接收完成， SSPBUF 满
0= 接收没有完成，SSPBUF 为空

图注：			
R= 可读位	W= 可写位	U= 未实现位，读作 0	
- n= 上电复位时的值	1= 置位	0= 清零	x= 未知

寄存器 15-2： SPI 模式下的 MSSP 控制寄存器 1：SSPCON1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL**：写冲突检测位 （仅在发送模式下）

1= 仍在发送前一字时，有数据要写入 SSPBUF 寄存器 （该位必须用软件清零）

0= 未发生冲突

bit 6 **SSPOV**：接收溢出标识位

SPI 从动模式：

1=SSPBUF 仍保存前一数据时，接收到新的字节。如果溢出，SSPSR 中的数据会丢失。溢出只会在从动模式下发生。即使只是发送数据，用户也必须读 SSPBUF，以避免溢出。（该位必须用软件清零）

0= 没溢出

注： 在主控模式下，溢出位不会被置 1，因为每次收发新数据，开始都要写入 SSPBUF 寄存器。

bit 5 **SSPEN**：同步串行口使能位

1= 使能串行口，并配置 SCK、 SDO、 SDI 和 $\overline{SS}$ 作为串行口引脚

0= 禁止串行口，并配置这些引脚为 I/O 口引脚

注： 当该位为 1 而使能时，应正确配置相应的引脚的输入输出。

bit 4 **CKP**：时钟极性选择位

1= 高电平为时钟空闲状态

0= 低电平为时钟空闲状态

bit 3-0 **SSPM3:SSPM0**：同步串行口模式选择位

0101=SPI 从动模式，时钟 =SCK 引脚，禁用 $\overline{SS}$ 引脚控制，可将 $\overline{SS}$ 用作 I/O 引脚

0100=SPI 从动模式，时钟 =SCK 引脚，使能 SS 引脚控制

0011=SPI 主控模式，时钟 =TMR2 输出 /2

0010=SPI 主控模式，时钟 =Fosc/64

0001=SPI 主控模式，时钟 =Fosc/16

0000=SPI 主控模式，时钟 =Fosc/4

注： 只在 I^2C 模式中保存或实现的位组合在这里就不一一列出了。

图注：			
R= 可读位	W= 可写位	U= 未实现位，读作 0	
- n= 上电复位时的值	1= 置位	0= 清零	x= 未知

15.3.2 工作模式

在初始化 SPI 的时候，有几项必须指定。可在程序中通过设置相应的控制位（SSPCON1<5:0>）和 SSPSTAT<7:6> 来指定。这些控制位允许指定下列模式：

- 主控模式 （SCK 作为时钟输出）
- 从动模式 （SCK 作为时钟输入）
- 时钟极性 （SCK 的空闲状态）
- 输入数据的采样相位 （数据输出期间的中段或结束时）
- 时钟沿 （在 SCK 的上升沿 / 下降沿输出数据）
- 时钟速率 （仅在主控模式下）
- 从动选择模式 （仅在从动模式下）

MSSP 模块由一个发送 / 接收移位寄存器（SSPSR）和一个缓冲寄存器 （SSPBUF）组成。 SSPSR 对器件的输入数据和输出数据进行移位，高位在前。在数据接收完毕前，SSPBUF 保存上次写入 SSPSR 的数据。一旦 8 位数据接收完毕，该字节被移入 SSPBUF 寄存器。同时缓冲区满检测位 BF （SSPSTAT<0>）和中断标志位 SSPIF 置 1。这种将接收到的数据进行双重缓冲的方式（SSPBUF），允许读取刚接收到的数据之前，开始接收下一个字节。在数据发送 / 接收期间，任何试图写 SSPBUF 寄存器的操作都会被忽略，并将写冲突检测位 WCOL（SSPCON<7>）置 1。用户必须用软件将 WCOL 位清零，才能判断之后对 SSPBUF 寄存器的写操作成功与否。

为确保应用软件能接收有效数据，应该在要传输的新数据写入 SSPBUF 之前，读取 SSPBUF 中现有的数据。缓冲区满标志位 BF（SSPSTAT<0>）用于表示 SSPBUF 是否已经载入了接收到的数据（发送完成）。当 SSPBUF 中的数据被读取后，BF 位即被清零。如果 SPI 仅仅作为一个发送器，则不必理会这一位。通常，MSSP 中断用于判断发送和接收何时完成。必须对 SSPBUF 进行读和 / 或写操作。如果不使用中断方法，可以进行软件查询来确保没有发生写冲突。例 15-1 表示数据发送模式下载入 SSPBUF （SSPSR）。

SSPSR 不能直接读写，存取只能通过对 SSPBUF 寄存器进行寻址的方式。另外， MSSP 状态寄存器（SSPSTAT）可用来表示各种状态条件。

例 15-1： 载入 SSPBUF （SSPSR）寄存器

```
LOOP BTFSS SSPSTAT, BF    ;Has data been received(transmit complete)?
     BRA   LOOP           ;No
     MOVF  SSPBUF, W      ;WREG reg=contents of SSPBUF
     MOVWF RXDATA         ;Save in user RAM, if data is meaningful
     MOVF  TXDATA, W      ;W reg=contents of TXDATA
     MOVWF SSPBUF         ;New data to xmit
```

15.3.3 使能 SPI I/O

要使能串行口，SSP 使能位 SSPEN（SSPCON1<5>）必须置 1。要复位或重新配置 SPI 模式，先将 SSPEN 位清零，重新初始化 SSPCON 寄存器，然后把 SSPEN 位置 1。这将配置 SDI、SDO、SCK 和 SS 引脚为串行口引脚。要使用这些引脚作为串行口，必须在程序中正确设置这些引脚的数据方向位（在 TRIS 寄存器中）。即：

- SDI 由 SPI 模块自动控制
- SDO 必须清零 TRISC<5> 位
- SCK （主控模式）必须清零 TRISC<3> 位
- SCK （从动模式）必须将 TRISC<3> 位置 1
- $\overline{\text{SS}}$ 必须将 TRISC<4> 位置 1

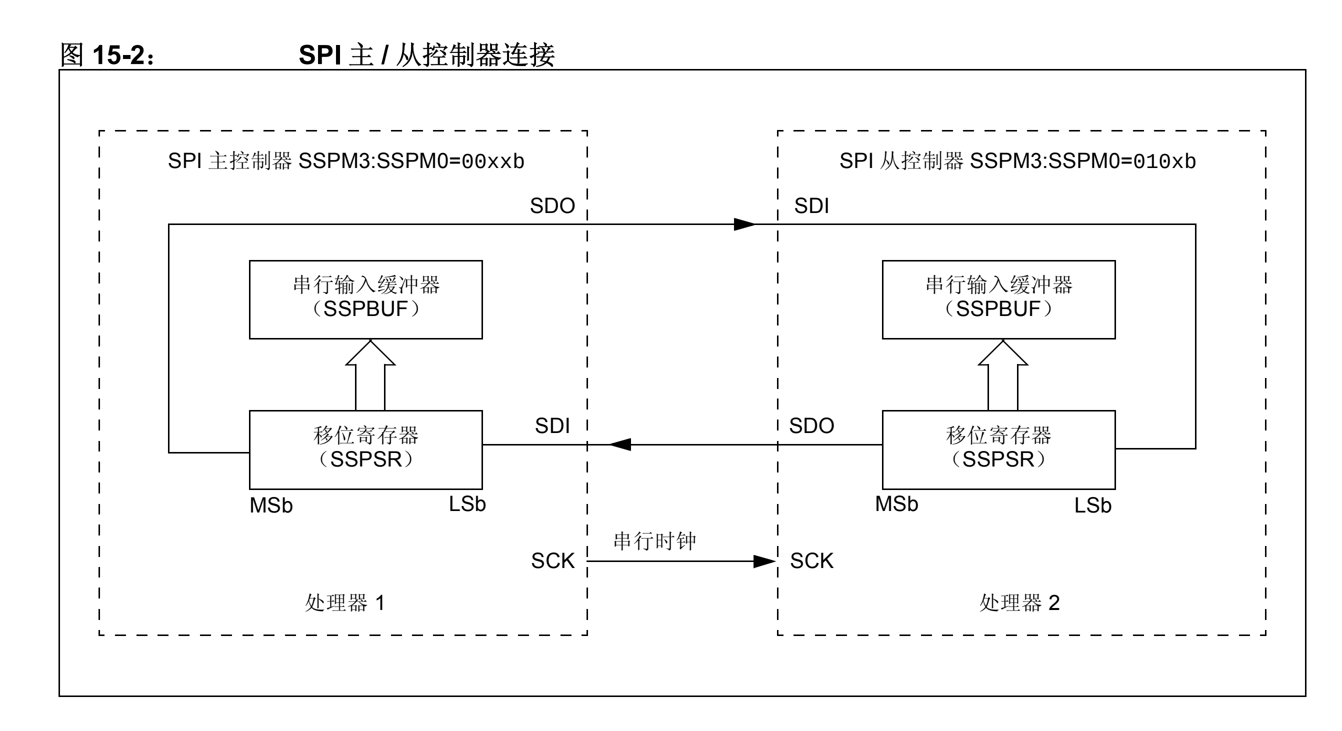
对于不需要的串行口功能，可以在程序中将相应的数据方向寄存器（TRIS）置反，用于其他用途。

15.3.4 典型连接

图 15-2 给出两个单片机之间的典型连接。主控制器（处理器 1）通过发送 SCK 信号来启动数据传输。两个移位寄存器中的数据都是在程序中指定的时钟沿上移出，并在下一个相反的时钟沿锁存。应该在程序中为两个处理器指定相同的时钟极性（Clock Polarity，CKP），这样两个控制器才能同时接发数据。数据是否有效取决于应用软件。这就有三种数据发送情境：

- 主控制器发送数据－从控制器发送无效数据
- 主控制器发送数据－从控制器发送数据
- 主控制器发送无效数据－ 从控制器发送数据

图 15-2： SPI 主 / 从控制器连接



15.3.5 主控模式

主控制器可以随时启动数据传输，因为它控制 SCK。主控制器根据软件协议确定从控制器（处理器 2，图 15-2）应在何时广播数据。

在主控模式下，数据一旦写入 SSPBUF 寄存器就开始发送或接收。如果 SPI 仅作为接收器，则可以禁止 SDO 输出（通过程序将其定义为输入口）。SSPSR 寄存器继续按程序指定的时钟速率，对 SDI 引脚上的信号执行移入。每收到一个字节，就将其装入 SSPBUF 寄存器，就象接收到的普通字节一样（中断和状态位相应置 1）。在“在线活动监视器”模式下工作的接收器应用中，这是很有用的。

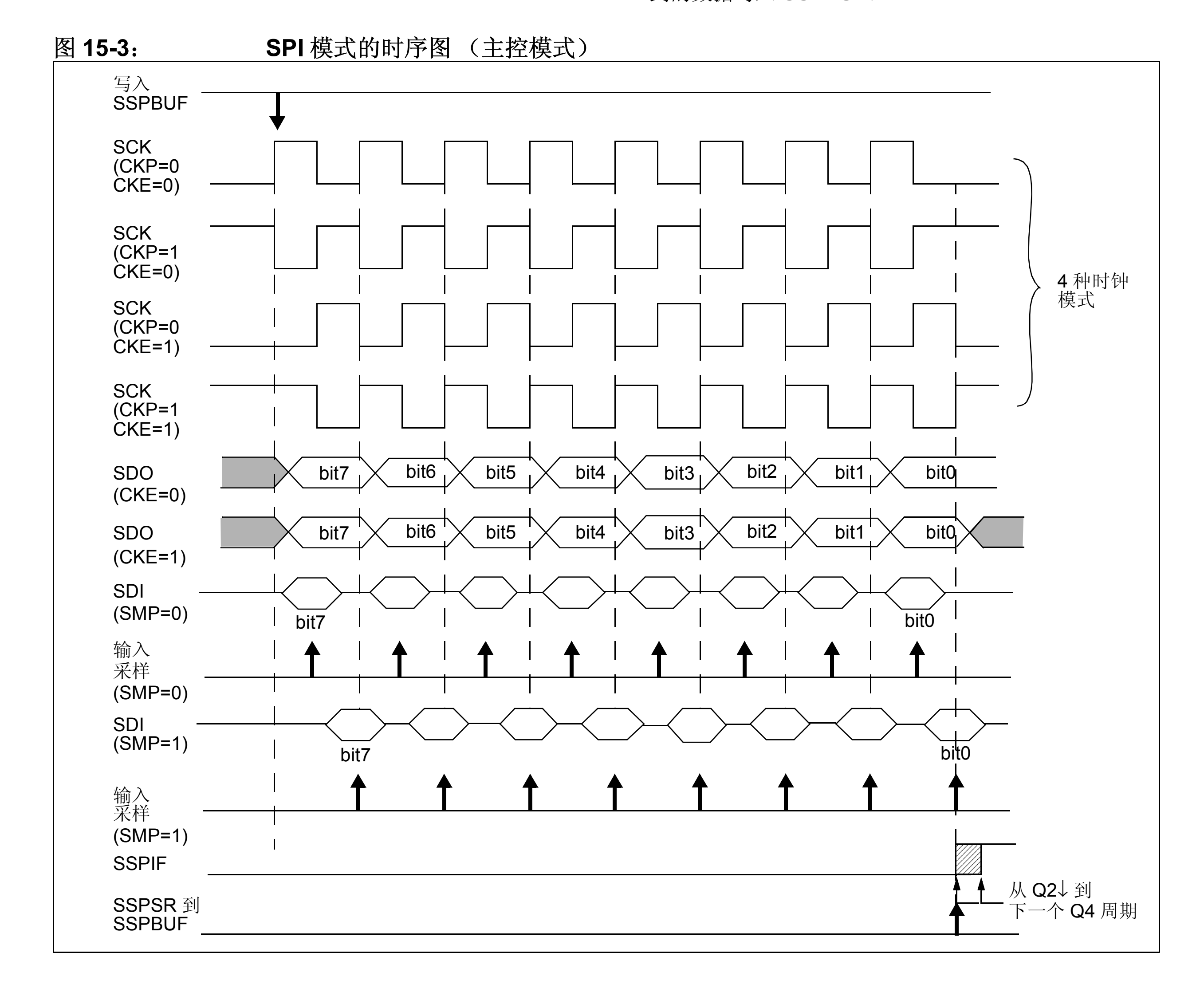
时钟极性可以通过在程序中正确地指定 CKP 位（SSPCON1<4>）来选择。图 15-3、图 15-5和图 15-6 将给出 SPI 通讯时序图，其中最先发送的是 MSB。在主控模式下，SPI 时钟速率（比特率）可以由用户通过程序指定为下列之一：

- FOSC/4 （即 TCY）
- FOSC/16 （即 4 • TCY）
- FOSC/64 （即 16 • TCY）
- Timer2 输出速率 /2

这里允许最大数据速率是 10.00 Mbps（当频率为 40 MHz 时）。

图 15-3 给出了主控模式的时序图。当 CKE 位置 1 时，SDO 数据在 SCK 的时钟沿处有效。图中所示输入采样的变化由 SMP 位的状态决定。图中指出了何时将接收到的数据写入 SSPBUF。

图 15-3： SPI 模式的时序图 （主控模式）



15.3.6 从动模式

在从动模式下，当 SCK 引脚上有外部时钟脉冲时发送 / 接收数据。当最后一位数据锁存后，中断标志位 SSPIF 置 1。

在从动模式下，外部时钟来自于 SCK 引脚上的外部时钟源。外部时钟的高低电平必须满足电气规范中规定的最小脉宽。

在休眠模式下，从控制器仍可发送 / 接收数据。当收到一个字节时，器件从休眠状态中唤醒。

15.3.7 从动选择同步

$\overline{SS}$ 引脚允许同步从动模式。 SPI 必须工作在从动模式下，并使能 SS 引脚控制（SSPCON1<3:0>=04h）。为使 SS 引脚作为输入端，不可使此引脚低电平驱动。数据锁存必须为高电平。当 $\overline{SS}$ 引脚为低电平时，使能数据的发送和接收，同时 SDO 引脚被驱动。当 $\overline{SS}$ 引脚为高电平时，即使是在数据的发送过程中，SDO 引脚也不再被驱动，而是变成悬空输出。根据应用的需要，可在 SDO 引脚上外接上拉 / 下拉电阻。

> **注 1：** 当 SPI 工作在从动模式下，并且 $\overline{SS}$ 引脚控制使能（SSPCON<3:0>=0100）时，如果 SS 引脚置为 VDD 电平，将使 SPI 模块复位。
>
> **2：** 如果 SPI 工作在从动模式下并且 CKE 置 1，则必须使能 SS 引脚控制。

SPI 模块复位时，位计数器将被清零。这可由强制 $\overline{SS}$ 引脚为高电平或清零 SSPEN 位来实现。

将 SDO 引脚和 SDI 引脚相连，可以仿真二线制通讯。当 SPI 需要作为接收器工作时，可将 SDO 引脚配置为输入端。这会禁止从 SDO 发送数据。而 SDI 一直作为输入端 （SDI 功能），因为它不会引起总线冲突。

图 15-4： 从动同步时序图

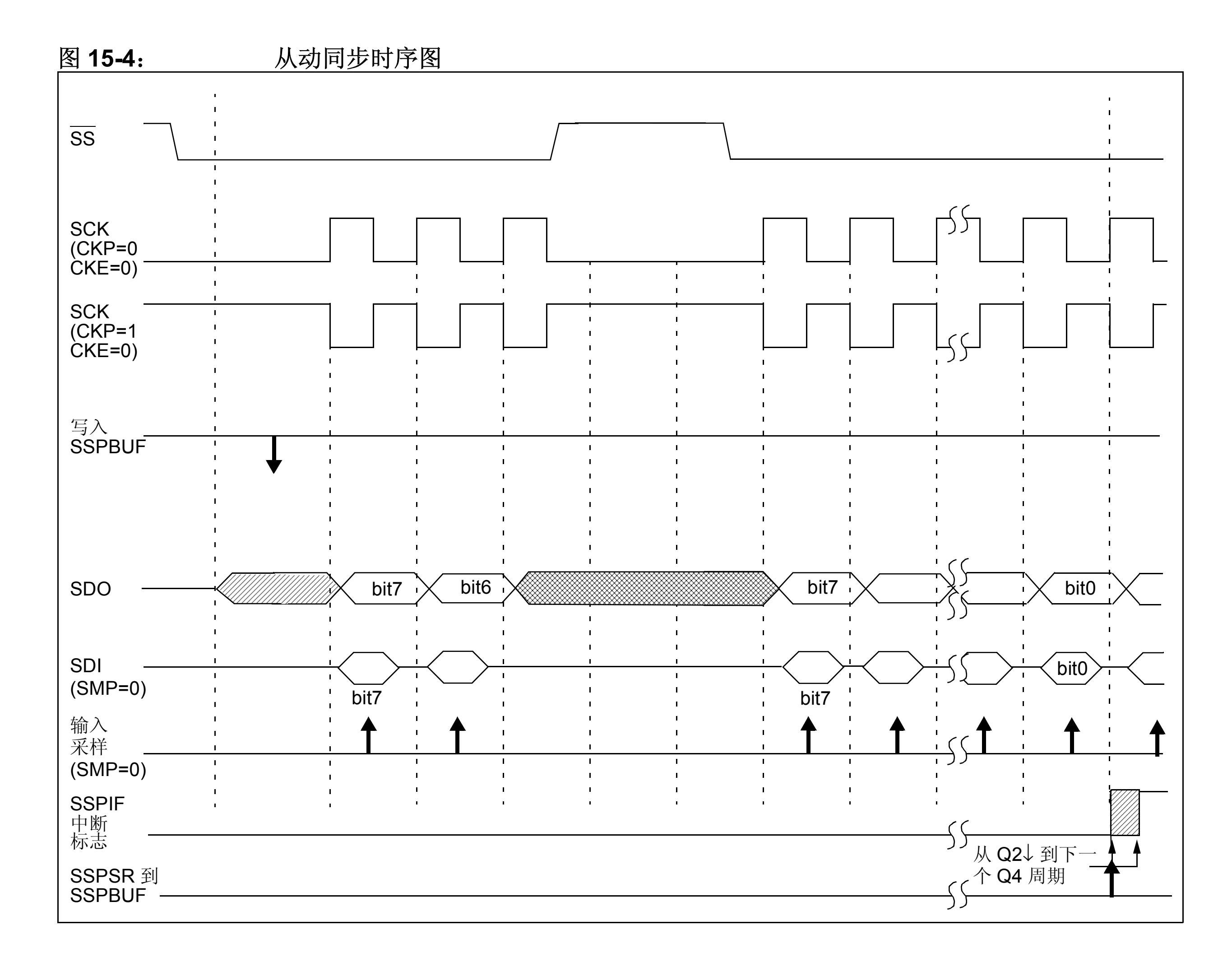


图 15-5： SPI 模式的时序图 （从动模式， CKE=0）

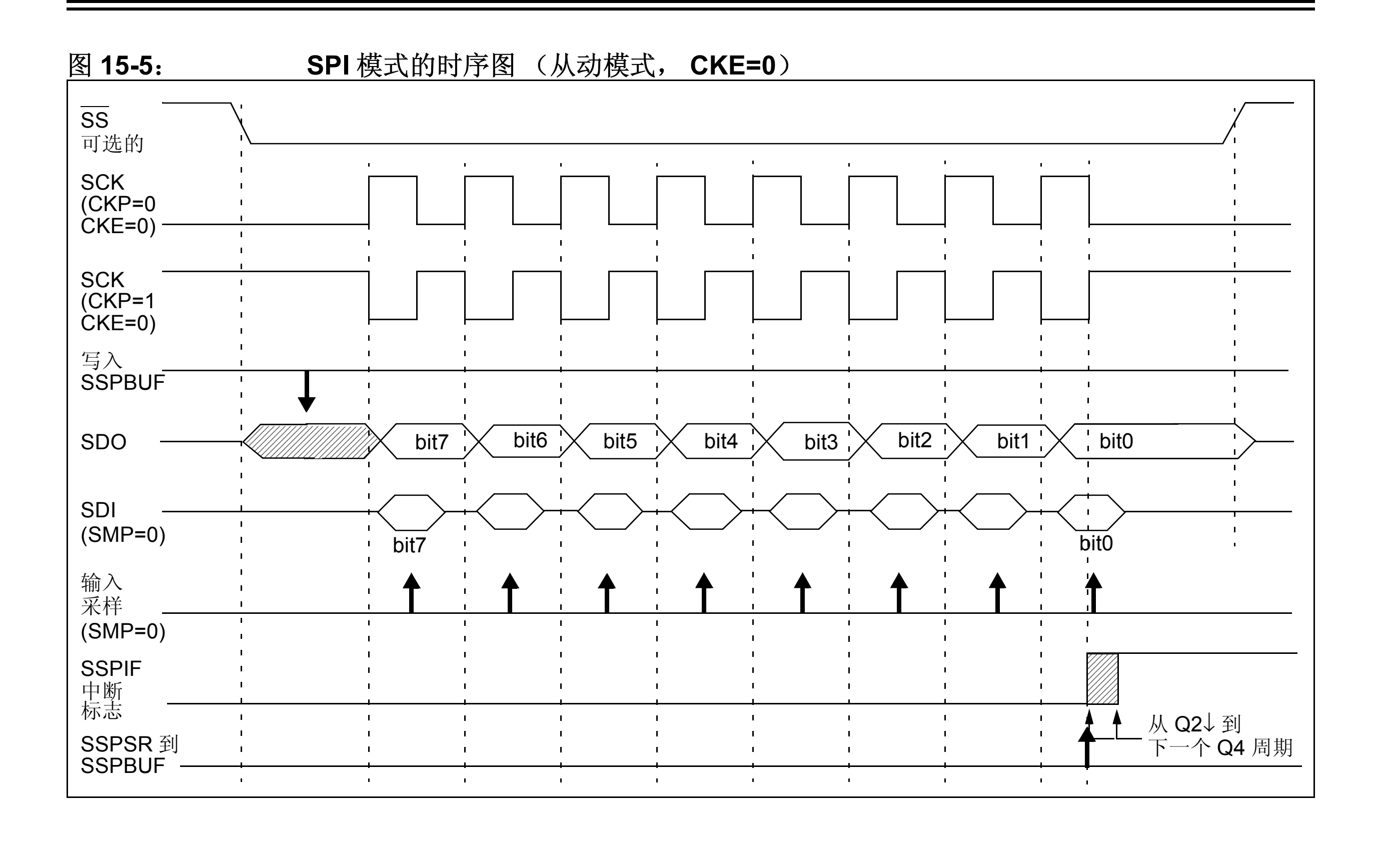
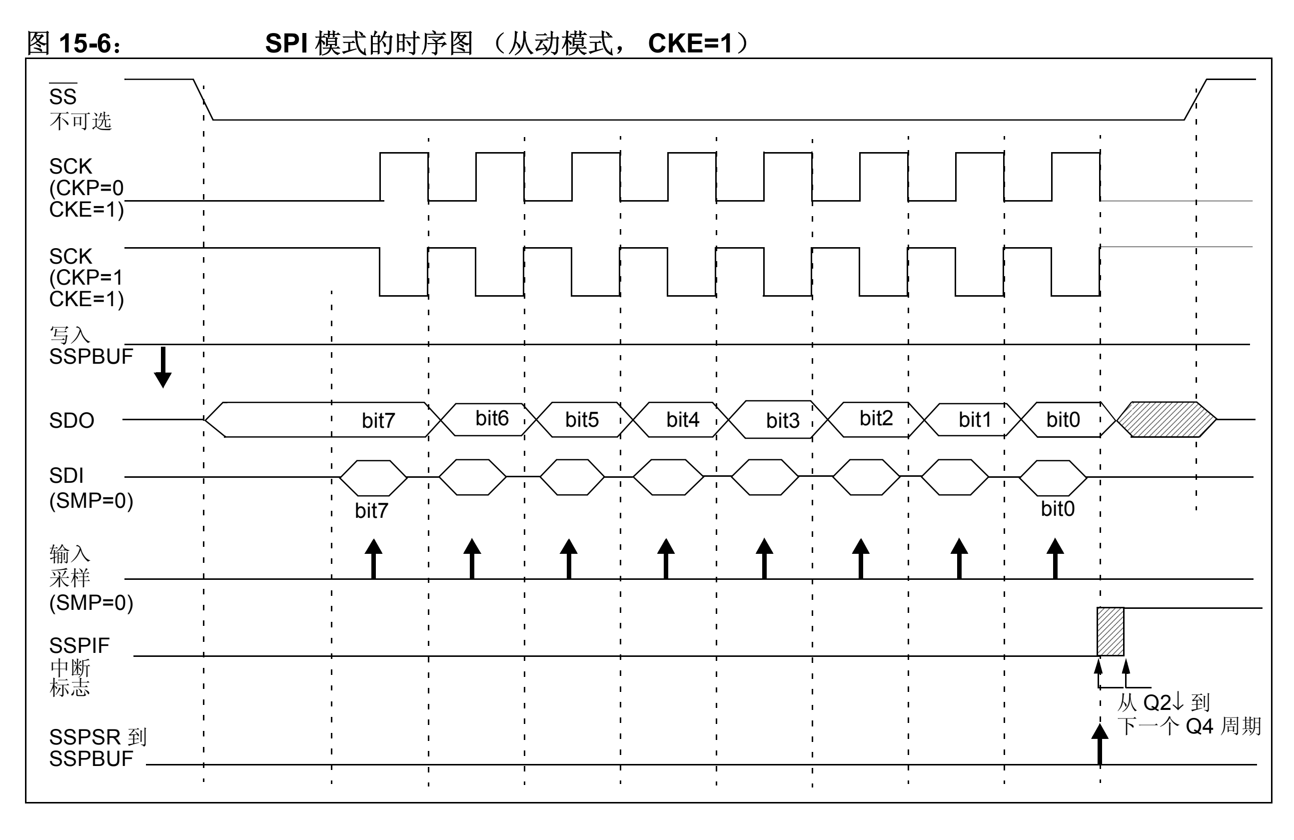


图 15-6： SPI 模式的时序图 （从动模式， CKE=1）



15.3.8 休眠状态下的操作

在主控模式下，在休眠状态时所有模块的时钟都停止，在将器件从休眠状态唤醒之前，发送 / 接收将一直处于休眠。在器件恢复正常工作模式后，模块将继续发送 / 接收数据。

在从动模式下， SPI 发送 / 接收移位寄存器与器件异步工作。这样，虽然器件处于休眠状态，但仍可将数据移入 SPI 发送 / 接收移位寄存器。当接收到全部 8 位后，MSSP 中断标志位将置 1，并且如果中断使能的话，将从休眠状态唤醒器件。

15.3.9 复位的影响

复位可以禁用 MSSP 模块并终止当前传输。

15.3.10 总线模式兼容性

表 15-1 中所示是标准 SPI 模式之间的兼容性及 CKP 与 CKE 控制位的状态。

表 15-1： SPI 总线模式

标准 SPI 模式术语	控制位状态	
	CKP	CKE
0, 0	0	1
0, 1	0	0
1, 0	1	1
1, 1	1	0

还有一个 SMP 位用来控制数据的采样时间。

表 15-2： 与 SPI 工作相关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	所有其他复位的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TRISC	PORTC 数据方向寄存器								1111 1111	1111 1111
SSPBUF	同步串行口接收缓冲器 / 发送寄存器								xxxx xxxx	uuuu uuuu
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
TRISA	—	PORTA 数据方向寄存器							-111 1111	-111 1111
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000

图注： x= 未知，u= 不变， -= 未实现，读作 0。阴影单元表示 MSSP 在 SPI 模式下未使用。

注 1：PIC18F2x2 器件保留 PSPIF、 PSPIE 和 PSPIP 位；始终将这些位清零。

15.4 I²C 模式

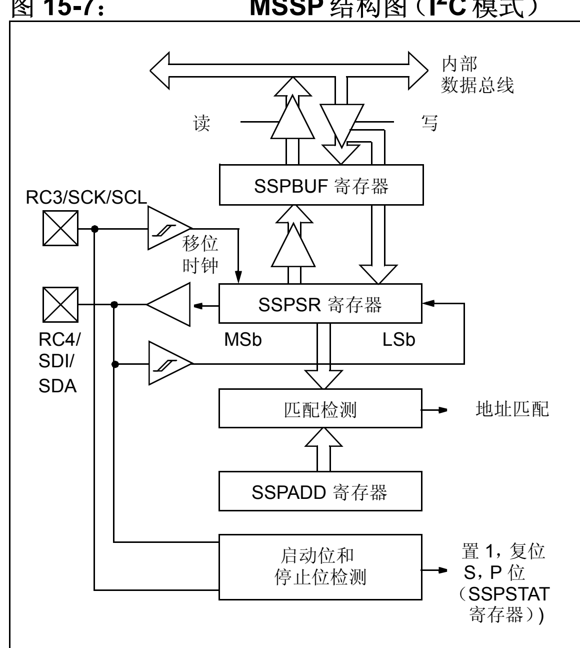
MSSP 模块工作在 I²C 模式时，可以完全实现所有主控和从动模式 （包括全局呼叫支持），并且通过硬件提供启动位中断和停止位中断以确定空闲总线 （多主控模式）。 MSSP 模块实现了标准模式规范，以及 7 位寻址和 10 位寻址。

有两个引脚用于数据传输。

- 串行时钟 （Serial Clock， SCL）— RC3/SCK/SCL
- 串行数据 （Serial Data， SDA）— RC4/SDI/SDA

用户必须通过 TRISC<4:3> 位将这些引脚配置为输入引脚或输出引脚。

图 15-7： MSSP 结构图（I²C 模式）



15.4.1 寄存器

MSSP 模块有六个寄存器用于 I²C 工作模式。它们是:

- MSSP 控制寄存器 1 （SSPCON1）
- MSSP 控制寄存器 2 （SSPCON2）
- MSSP 状态寄存器 （SSPSTAT）
- 串行接收 / 发送缓冲器 （SSPBUF）
- MSSP 移位寄存器 （SSPSR）－不可直接存取
- MSSP 地址寄存器 （SSPADD）

SSPCON、SSPCON2 和 SSPSTAT 是在 I²C 模式下工作的控制寄存器和状态寄存器。SSPCON 和 SSPCON2 寄存器是可读写的。 SSPSTAT 的低六位是只读的。SSPSTAT 的高两位是可读写的。

SSPSR 是用来将数据移入或移出的移位寄存器。SSPBUF 是缓冲寄存器，可用于读写数据字节。

在 I²C 从动模式下配置 SSP 时， SSPADD 寄存器将保存从动器件地址。在主控模式下配置 SSP 时，SSPADD 的低七位用作波特率发生器的重载值。

接收时，SSPSR 和 SSPBUF 共同构成了一个双重缓冲接收器。当 SSPSR 接收完整的字节后，将其送入 SSPBUF，同时将中断标志位 SSPIF 置 1。

在发送中，SSPBUF 并不是双重缓冲的。写入 SSPBUF 的同时将写入 SSPBUF 和 SSPSR。

寄存器 15-3： I²C 模式下的 MSSP 状态寄存器：SSPSTAT

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF
bit 7							bit 0

bit 7 **SMP**：转换率控制位

在主控或从动模式下：

1= 在标准速度模式 （100 kHz 和 1 MHz）下，禁用转换率控制
0= 在高速模式 （400 kHz）下，使能转换率控制

bit 6 **CKE**：SMBus 选择位

在主控或从动模式下：

1= 使能 SMBus 特定输入
0= 禁用 SMBus 特定输入

bit 5 **D/$\overline{\text{A}}$**：数据 / $\overline{\text{地址}}$位

在主控模式下：

保留

在从动模式下：

1= 表明接收或发送的最后字节是数据
0= 表明接收或发送的最后字节是地址

bit 4 **P**：停止位

1= 表示最近检测到停止位
0= 表示未检测到停止位

注： 复位时和 SSPEN 清零时，该位清零。

bit 3 **S**：启动位

1= 表示最后检测到启动位
0= 表示未检测到启动位

注： 复位时和 SSPEN 清零时，该位清零。

bit 2 **R/$\overline{\text{W}}$**：读 / 写位信息 （仅在 I²C 模式下）

在从动模式下：

1= 读
0= 写

注： 该位用来保存在最近一次地址匹配后的 R/W 位信息。仅从本机地址与接收地址匹配开始，到下一个启动位、停止位或无 ACK 位时，该位有效。

在主控模式下：

1= 正在进行发送
0= 未进行发送

注： 该位与 SEN、 RSEN、 PEN、 RCEN 或 ACKEN 进行 “或”运算的结果表示 MSSP 是否处在空闲模式。

bit 1 **UA**：更新地址 （仅在 10 位从动模式下）

1= 用户需要更新 SSPADD 寄存器中的地址
0= 不需要更新地址

bit 0 **BF**：缓冲器满状态位

在发送模式下：

1= 接收完成， SSPBUF 满
0= 接收没有完成， SSPBUF 空

在接收模式下：

1= 数据正在发送 （不包括 $\overline{\text{ACK}}$ 位和停止位）， SSPBUF 满
0= 数据发送完毕 （不包括 ACK 位和停止位）， SSPBUF 空

图注：			
R= 可读位	W= 可写位	U= 未实现位，读作 0	
- n= 上电复位时的值	1= 置位	0= 清零	x= 未知

寄存器 15-4： I^2C 模式下的 MSSP 的控制寄存器 1：SSPCON1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL**：写冲突检测位

在主控发送模式下：
1= 当 I^2C 不满足开始发送数据的条件时有数据要写入 SSPBUF 寄存器 （该位必须用软件清零）
0= 未发生冲突

在从动发送模式下：
1= 正在发送前一个字时又有数据写入 SSPBUF 寄存器
（该位必须用软件清零）
0= 未发生冲突

在接收模式 （主控或从动模式）下：
这是一个 “忽略”位

bit 6 **SSPOV**：接收溢出标识位

在接收模式下：
1= 当 SSPBUF 寄存器中仍保存前一个字节时收到了下一个字节
（该位必须用软件清零）
0= 没有溢出

在发送模式下：
这在发送模式下是 “忽略”位

bit 5 **SSPEN**：同步串行口使能位
1= 使能串行口，并配置 SDA 和 SCL 引脚为串行口引脚
0= 禁止串行口，并配置这些引脚为 I/O 口引脚

注： 当该位为 1 而使能时，必须正确配置 SDA 和 SCL 引脚为输入引脚或输出引脚。

bit 4 **CKP**：SCK 释放控制位

在从动模式下：
1= 释放时钟
0= 保持时钟线为低电平 （时钟延长），用于保证数据的建立时间

在主控模式下：
在此模式下未使用

bit 3-0 **SSPM3:SSPM0**：同步串行口模式选择位
1111=I^2C 从动模式， 10 位地址，允许启动位中断和停止位中断
1110 =I^2C 从动模式， 7 位地址，允许启动位中断和停止位中断
1011=I^2C 固件控制的主控模式 （从动空闲）
1000=I^2C 主控模式，时钟 =FOSC / (4 * (SSPADD+1))
0111=I^2C 从动模式， 10 位地址
0110=I^2C 从动模式， 7 位地址

注： 只在 SPI 模式下保存或实现的位组合这里就不一一列出了。

图注：			
R= 可读位	W= 可写位	U= 未实现位，读作 0	
- n= 上电复位时的值	1= 置位	0= 清零	x= 未知

寄存器 15-5： **I²C 模式下的 MSSP 控制寄存器 2：SSPCON2**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

bit 7 **GCEN**：全局呼叫使能位 （仅从动模式）
1=SSPSR 接收到全局呼叫地址 （0000h）时使能中断
0= 禁止全局呼叫地址

bit 6 **ACKSTAT**：应答状态位 （仅主控发送模式）
1= 没有收到来自从动器件的应答
0= 收到来自从动器件的应答

bit 5 **ACKDT**：应答数据位 （仅主控接收模式）
1= 不应答
0= 应答

注： 用户在接收结束时启动一个应答序列，同时发送该值。

bit 4 **ACKEN**：应答序列使能位 （仅主控接收模式）
1= 在 SDA 和 SCL 引脚启动应答序列，发送 ACKDT 数据位。
由硬件自动清零。
0= 应答序列空闲

bit 3 **RCEN**：接收使能位 （仅主控模式）
1= 使能 I²C 接收模式
0= 接收空闲

bit 2 **PEN**：停止条件使能位 （仅主控模式）
1= 在 SDA 和 SCL 引脚启动停止条件。由硬件自动清零。
0= 停止条件空闲

bit 1 **RSEN**：重复启动条件使能位 （仅主控模式）
1= 在 SDA 和 SCL 引脚启动重复启动条件。
由硬件自动清零。
0= 重复启动条件空闲

bit 0 **SEN**：启动条件使能 / 延长使能位
在主控模式下：
1= 在 SDA 和 SCL 引脚启动启动条件。由硬件自动清零。
0= 启动条件空闲
在从动模式下：
1= 为从动发送和从动接收使能时钟延长 （已使能延长）
0= 仅为从动发送使能时钟延长 （旧模式）

注： 对于 ACKEN、RCEN、PEN、RSEN 和 SEN 位：如果 I²C 模块不在空闲模式，不将该位置 1（不支持并行操作），也不对 SSPBUF 进行写操作（或者禁止向 SSPBUF 写）。

图注：			
R= 可读位	W= 可写位	U= 未实现位，读作 0	
- n= 上电复位时的值	1= 置位	0= 清零	x= 未知

15.4.2 工作模式

将 MSSP 使能位 SSPEN （SSPCON<5>）置 1，可使能 MSSP 模块功能。

SSPCON1 寄存器可控制 I^2C 的工作模式。可通过四个模式选择位 （SSPCON<3:0>）来选择下列 I^2C 模式之一：

- I^2C 主控模式，时钟 =OSC/4 (SSPADD +1)
- I^2C 从动模式 （7 位地址）
- I^2C 从动模式 （10 位地址）
- I^2C 从动模式 （7 位地址），允许启动位和停止位中断功能
- I^2C 从动模式 （10 位地址），允许启动位和停止位中断功能
- I^2C 固件控制的主控操作，从动模式为空闲

任何 I^2C 模式的选择，在 SSPEN 置 1 后都会强制 SCL 和SDA引脚为漏极开路（假定通过程序将相应的TRISC 位置 1，使这些引脚成为输入引脚）。要确保此模块的正常工作，必须为SCL和SDA引脚提供外接上拉电阻。

15.4.3 从动模式

在从动模式下， SCL 和 SDA 引脚必须设置为输入（TRISC<4:3> 置 1）。在需要时（如在从动发送器情况下）， MSSP 模块会强行将输入状态改变为输出数据。

I^2C 从动模式硬件往往在地址匹配时发生中断。用户也可以通过模式选择位的设定来选择启动位或停止位中断。

当地址匹配时或在地址匹配后传送的数据被接收时，硬件会自动产生一个应答（ACK）脉冲，并把当时 SSPSR 寄存器中接收到的数据装入 SSPBUF 寄存器。

只要满足下列条件之一，MSSP 模块就不会产生此 $\overline{ACK}$ 脉冲：

- 缓冲器满位 BF （SSPSTAT<0>）在发送的数据被接收之前就置 1 了。
- 溢出位 SSPOV （SSPCON<6>）在发送的数据被接收之前就置 1 了。

在这种情况下，SSPSR寄存器的值不会载入SSPBUF，但是 SSPIF 位 （PIR1<3>）会置 1。 BF 位是通过读取 SSPBUF 寄存器清零的，而 SSPOV 位必须通过软件来清零。

为确保正常工作，SCL 时钟输入的高低电平必须满足最小脉宽。关于 I^2C 规范所规定的高低电平脉宽以及对 MSSP 模块的具体要求，请参见参数 100 和参数 101。

15.4.3.1 寻址

一旦 MSSP 模块被使能，它就等待满足启动条件。在启动条件发生后， 8 位数据被移入 SSPSR 寄存器。所有移入的位都是在时钟线 （SCL）的上升沿进行采样的。寄存器 SSPSR<7:1> 与 SSPADD 寄存器的值比较；这一地址比较在第 8 个时钟（SCL）脉冲下降沿进行。如果地址匹配，BF 和 SSPOV 位就会被清零，并完成下列操作：

1. SSPSR 寄存器值被装入 SSPBUF 寄存器。
2. 缓冲器满位 BF 被置 1。
3. 产生 $\overline{ACK}$ 脉冲。
4. 在 SCL 的第 9 个脉冲的下降沿时， MSSP 中断标志位 SSPIF （PIR1<3>）置 1 （如果使能中断，则产生中断）。

在 10 位地址模式下，从动器件需要接收两个地址字节。第一个地址字节的高 5 位 （MSb）指定这是否是一个 10 位地址。 R/$\overline{W}$ 位 （SSPSTAT<2>）必须指定为写操作，这样从动器件就会接收第二个地址字节。对于 10 位地址，第一个字节应该是 “`11110 A9 A8 0`”，其中 “A9”和 “A8”是 10 位地址的两个最高有效位。10 位地址的工作步骤如下，其中 7-9 步是针对从动发送器而言的：

1. 接收地址的第一个 （高）字节 （SSPIF 位、 BF 位和 UA 位 （SSPSTAT<1>）被置 1）。
2. 用地址的第二个 （低）字节更新 SSPADD 寄存器 （对 UA 位清零，同时释放 SCL 时钟线）。
3. 读 SSPBUF 寄存器 （BF 位请零），并将标志位 SSPIF 清零。
4. 接收地址的第二个 （低）字节 （SSPIF 位、 BF 位和 UA 位被置 1）。
5. 用地址的第一个 （高）字节更新 SSPADD 寄存器。如果匹配的话，就释放 SCL 时钟线，这将清零 UA 位。
6. 读 SSPBUF 寄存器 （BF 位清零），并将标志位 SSPIF 清零。
7. 接收重复的启动信号。
8. 接收地址的第一个 （高）字节 （SSPIF 位和 BF 位被置 1）。
9. 读 SSPBUF 寄存器 （BF 位请零），并将标志位 SSPIF 清零。

15.4.3.2 接收

当地址字节的 R/$\overline{W}$ 位清零而且出现了地址匹配时，SSPSTAT 寄存器的 R/$\overline{W}$ 位被清零。接收的地址被装入 SSPBUF 寄存器，且 SDA 保持低电平 （$\overline{ACK}$）。

当发生地址字节溢出时，不会产生应答脉冲 （$\overline{ACK}$）。溢出条件是指BF位（SSPSTAT<0>）置1，或者SSPOV 位 （SSPCON1<6>）置 1。

每个数据传输字节都会产生一个MSSP 中断。SSPIF 标志位 （PIR1<3>）必须用软件清零。 SSPSTAT 寄存器用于确定字节的状态。

如果 SEN 被使能（SSPCON1<0>=1），RC3/SCK/SCL 将在每个数据传输之后保持为低电平 （时钟延长）。必须通过将 CKP 位（SSPCON<4>）置 1 才能释放时钟。如需了解更多细节，请参阅第 15.4.4 节（ “时钟延长”）。

15.4.3.3 发送

当地址字节的 R/$\overline{W}$ 位置 1 且地址匹配时，SSPSTAT 寄存器的 R/$\overline{W}$ 位被置 1。接收到的地址将装入 SSPBUF 寄存器。$\overline{ACK}$ 脉冲在第 9 位上发送，同时不管 SEN 的值如何 （如需了解更多细节，请参阅第 15.4.4 节 “时钟延长”），RC3/SCK/SCL 引脚保持低电平。通过延长时钟，主控器件只有在从动器件准备好发送数据时才能决定另一个时钟脉冲。发送的数据必须装入SSPBUF 寄存器和 SSPSR 寄存器。这样，应将 CKP 位（SSPCON1<4>）置 1，以使能 RC3/SCK/SCL 引脚。在 SCL 输入的下降沿时移出 8 个数据位。这样可以确保 SCL 为高电平时 SDA 信号有效 （图 15-9）。

主接收器的 $\overline{ACK}$ 脉冲将在 SCL 输入的第 9 个脉冲上升沿锁存。如果 SDA 为高电平 （无 ACK 应答信号），则数据传输完成。在这种情况下，如果从动器件锁存了 ACK，将复位从动逻辑 （复位 SSPSTAT 寄存器），并且从动器件监视下一个启动位的发生。如果 SDA 为低电平 （有 ACK 应答信号），则必须将接下去要发送的数据装入 SSPBUF 寄存器。同样，必须将 CKP 位置 1 才能使能 RC3/SCK/SCL 引脚。

每个数据传输字节都会产生一个MSSP 中断。SSPIF 位必须用软件清零， SSPSTAT 寄存器用于确定字节的状态。 SSPIF 位在第 9 个时钟脉冲的下降沿被置 1。

图 15-8： I²C 从动模式时序， SEN=0 （接收， 7 位地址）

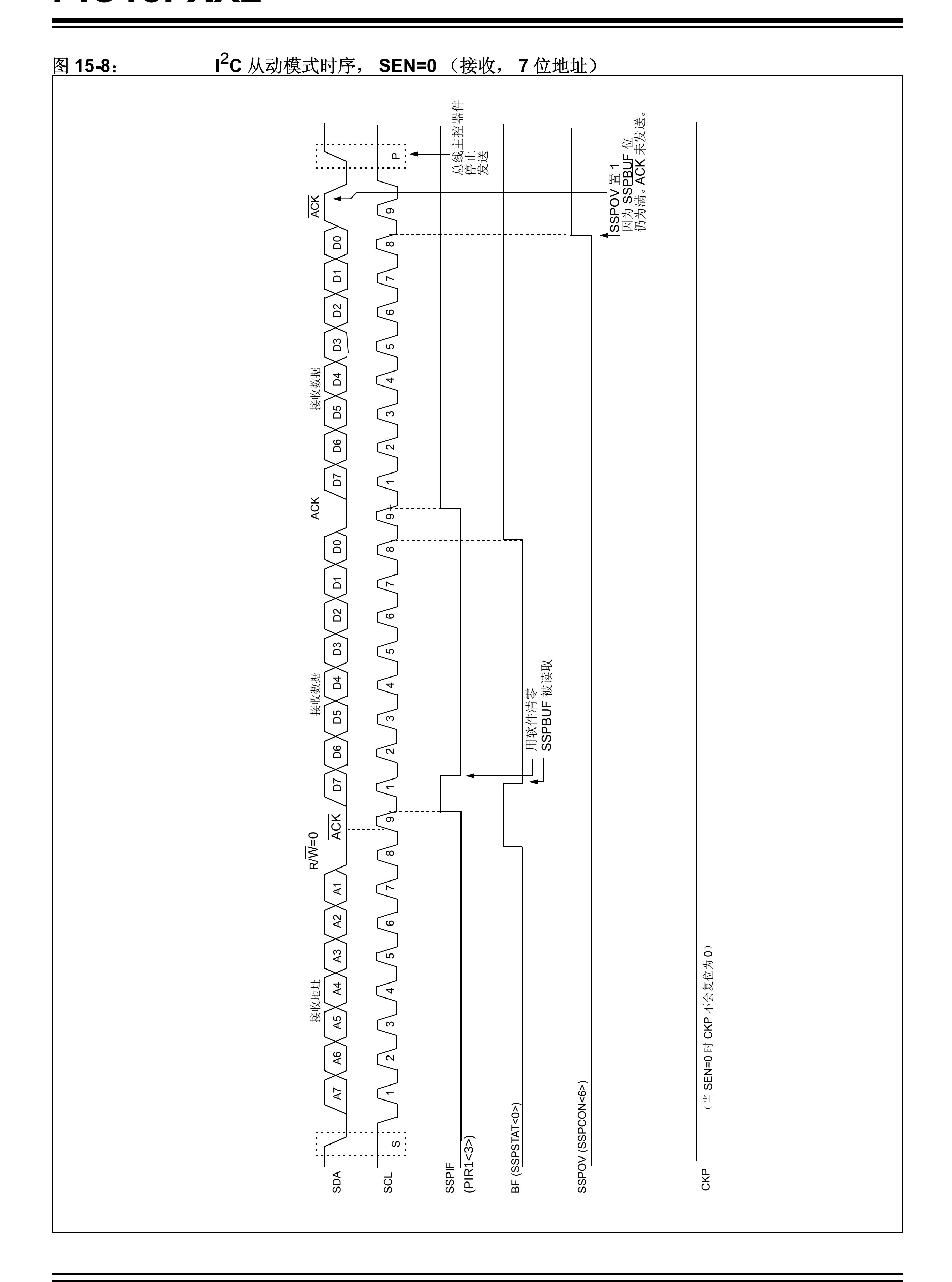


图 15-9： I^2C 从动模式时序 （发送， 7 位地址）

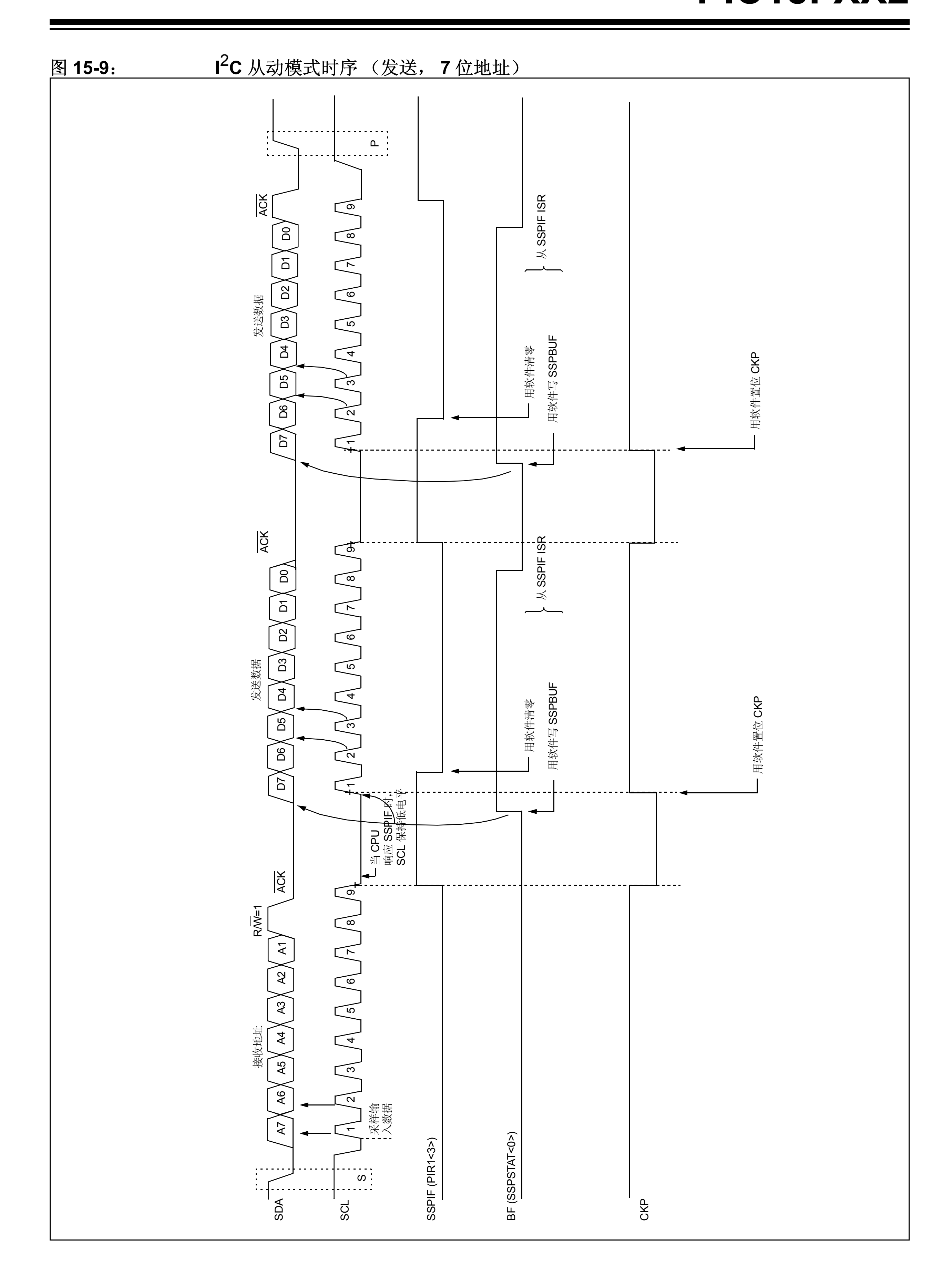


图 15-10： I^2C 从动模式时序， **SEN=0** （接收， **10** 位地址）

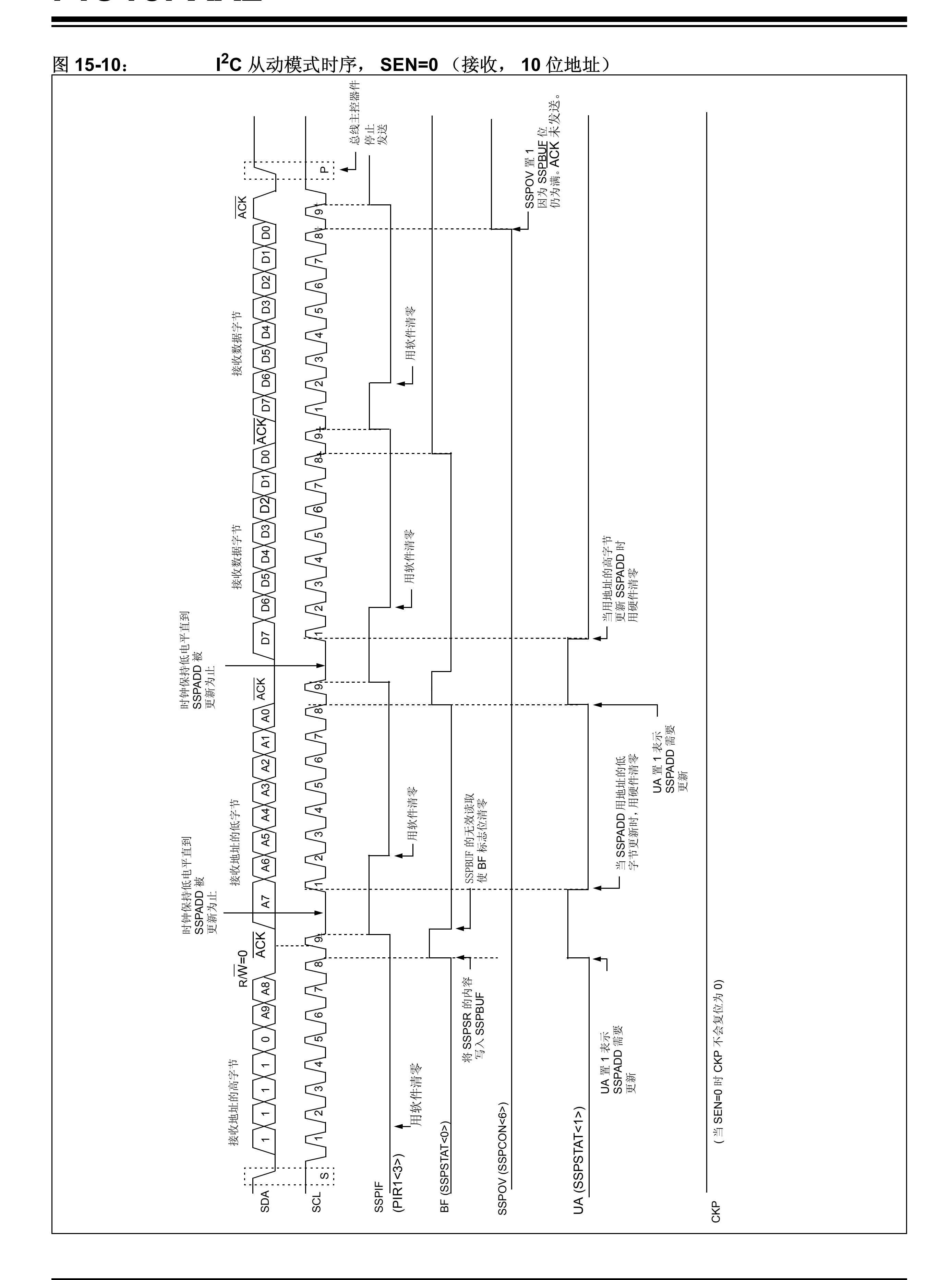
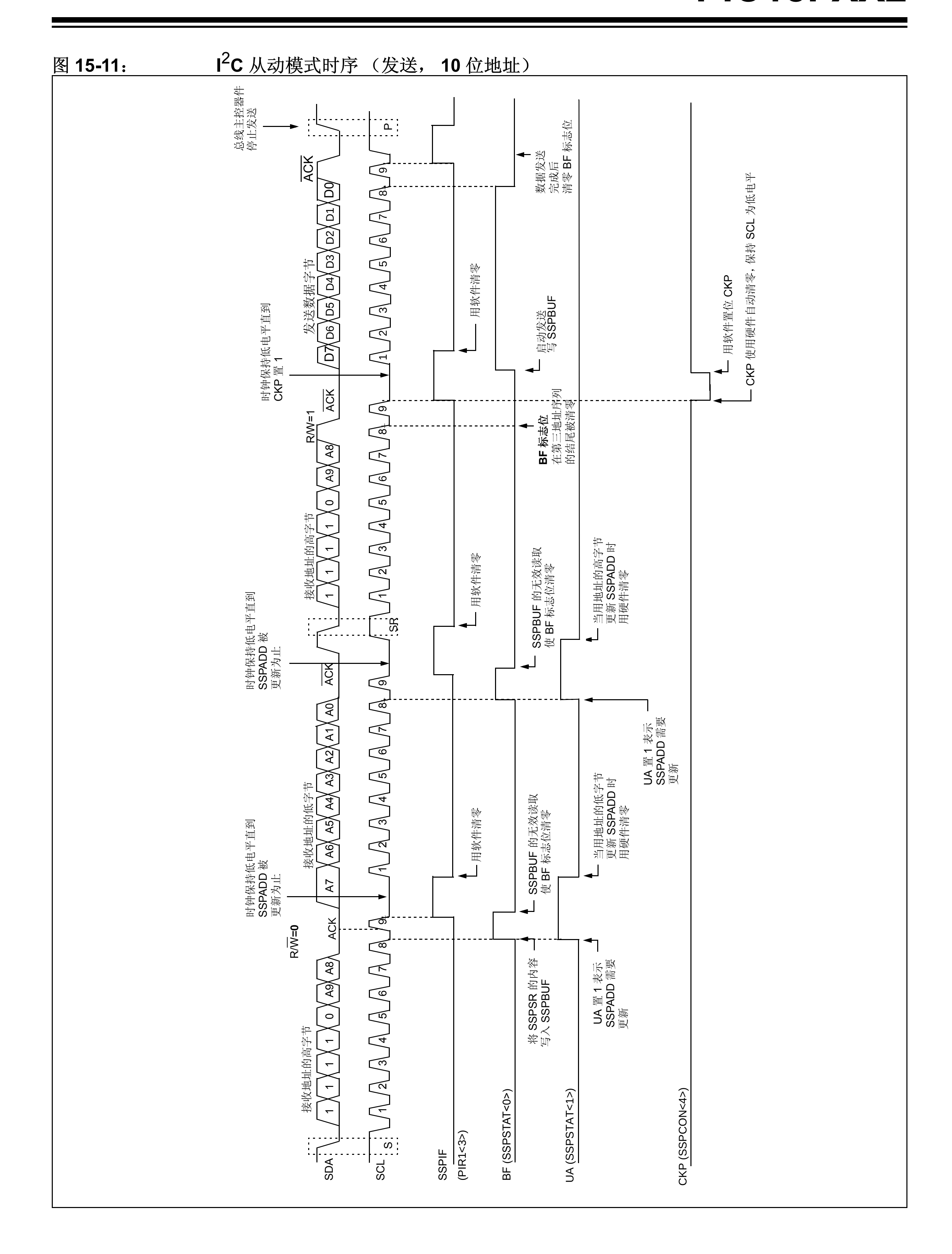


图 15-11： I^2C 从动模式时序 （发送， 10 位地址）



15.4.4 时钟延长

7位和10位从动模式都通过发送序列来实现自动时钟延长。

SEN 位（SSPCON2<0>）允许在接收过程中使能时钟延长。将 SEN 置 1 会导致在每个数据接收序列结束时 SCL 引脚保持在低电平。

15.4.4.1 7 位从动接收模式（SEN=1）的时钟延长

在 7 位从动接收模式下，如果在 $\overline{ACK}$ 序列结束处的第 9 个时钟下降沿 BF 位被置 1，则 SSPCON1 寄存器中的CKP位就会自动清零，强制SCL输出保持在低电平。CKP 位被清零会决定 SCL 为低电平。在允许继续接收之前，必须在用户的 ISR 中将 CKP 位置 1。保持 SCL 为低电平可以让用户在主控器件开始另一个接收序列之前，有时间对 ISR 做出响应并读取 SSPBUF 的内容。这将防止发生缓冲器溢出（见图 15-13）。

> 注 **1**：如果用户在第 9 个时钟的下降沿前读取了 SSPBUF 的内容，使得 BF 位清零， CKP 位就不会被清零，也不会产生时钟延长。
>
> **2**：不管 BF 位的状态如何， CKP 位都可以用软件置 1。用户在下一个接收序列开始之前，ISR 中清零 BF 位时应该小心，以免溢出。

15.4.4.2 10 位从动接收模式（SEN=1）的时钟延长

在 10 位从动接收模式中，在地址序列中自动产生时钟延长，但是 CKP 位不会被清零。在这期间，如果 UA 位在第 9 时钟之后置 1，时钟延长就启动了。UA 位在接收 10 位地址的高字节后置 1，然后接收 10 位地址中的低字节并将 R/$\overline{W}$ 位清零。在更新 SSPADD 时释放时钟。如 7 位模式中描述的那样，在每个数据接收序列中会产生时钟延长。

> **注：** 如果用户在第 9 个时钟下降沿之前查询 UA 位，并更新 SSPADD 寄存器以使 UA 位清零，而且没有提前读取 SSPBUF 寄存器使 BF 位清零，则 CKP 位的电平仍然不会被拉低。基于 BF 位状态的时钟延长仅在数据序列中发生，不会发生在地址序列中。

15.4.4.3 7 位从动发送模式的时钟延长

如果 BF 位被清零， 7 位从动发送模式将在第 9 个时钟的下降沿之后清零CKP位，以实现时钟延长。不管SEN位的状态如何，这种情况都会发生。

用户的 ISR 必须先将 CKP 位置 1 才可以继续发送。通过保持SCL为低电平，在主控器件开始另一个发送序列之前，用户将有时间响应 ISR 并装入 SSPBUF 的内容（见图 15-9）。

> 注 **1**：如果用户在第 9 个时钟下降沿之前就装入 SSPBUF 的内容，使 BF 位置 1， CKP 位就不会被清零，也不会产生时钟延长。
>
> **2**：不管 BF 位的状态如何， CKP 位都可以用软件置 1。

15.4.4.4 10 位从动发送模式的时钟延长

在10位从动发送模式中，在前两个地址序列中由UA位的状态来控制时钟延长，与10位从动接收模式一样。第三个地址序列在前两个地址序列之后，它包含 10 位地址的高位，且 R/$\overline{W}$ 位置 1。在执行完第三个地址序列之后，UA 位不置 1，此时模块配置为发送模式，BF 标志位控制时钟延长，就象在 7 位从动发送模式中一样（见图 15-11）。

15.4.4.5 时钟同步与 CKP 位

如果用户将 CKP 位清零，SCL 将强制输出为“0”。只有当采样到 SCL 输出为低电平时，将 CKP 位置 1 才使 SCL 输出变为低电平。如果用户试图将 SCL 变为低电平，在外部 I²C 主控器件将 SCL 拉低后， CKP 位才将 SCL 拉低。SCL 输出将保持低电平，直到 CKP 位置 1，且 I²C 总线上的所有其他器件不能将 SCL 电平拉高。这可以保证对 CKP 位的写操作不会违反 SCL 高电平的最小脉宽要求 （见图 15-12）。



图 15-12： 时钟同步时序

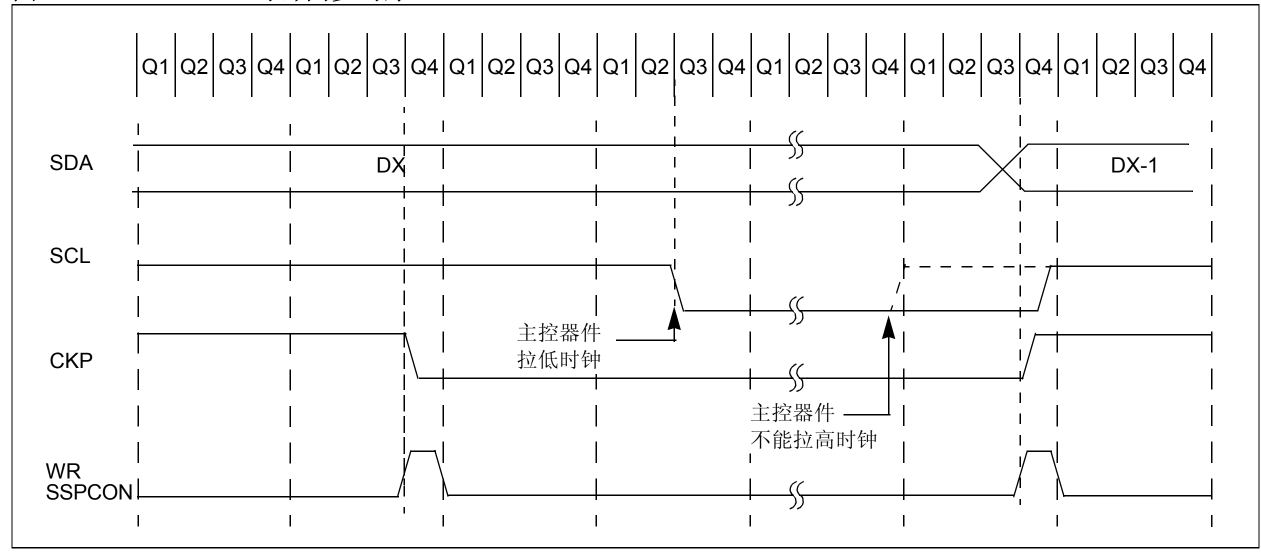


图 15-13： I²C 从动模式时序， SEN=1 （接收， 7 位地址）

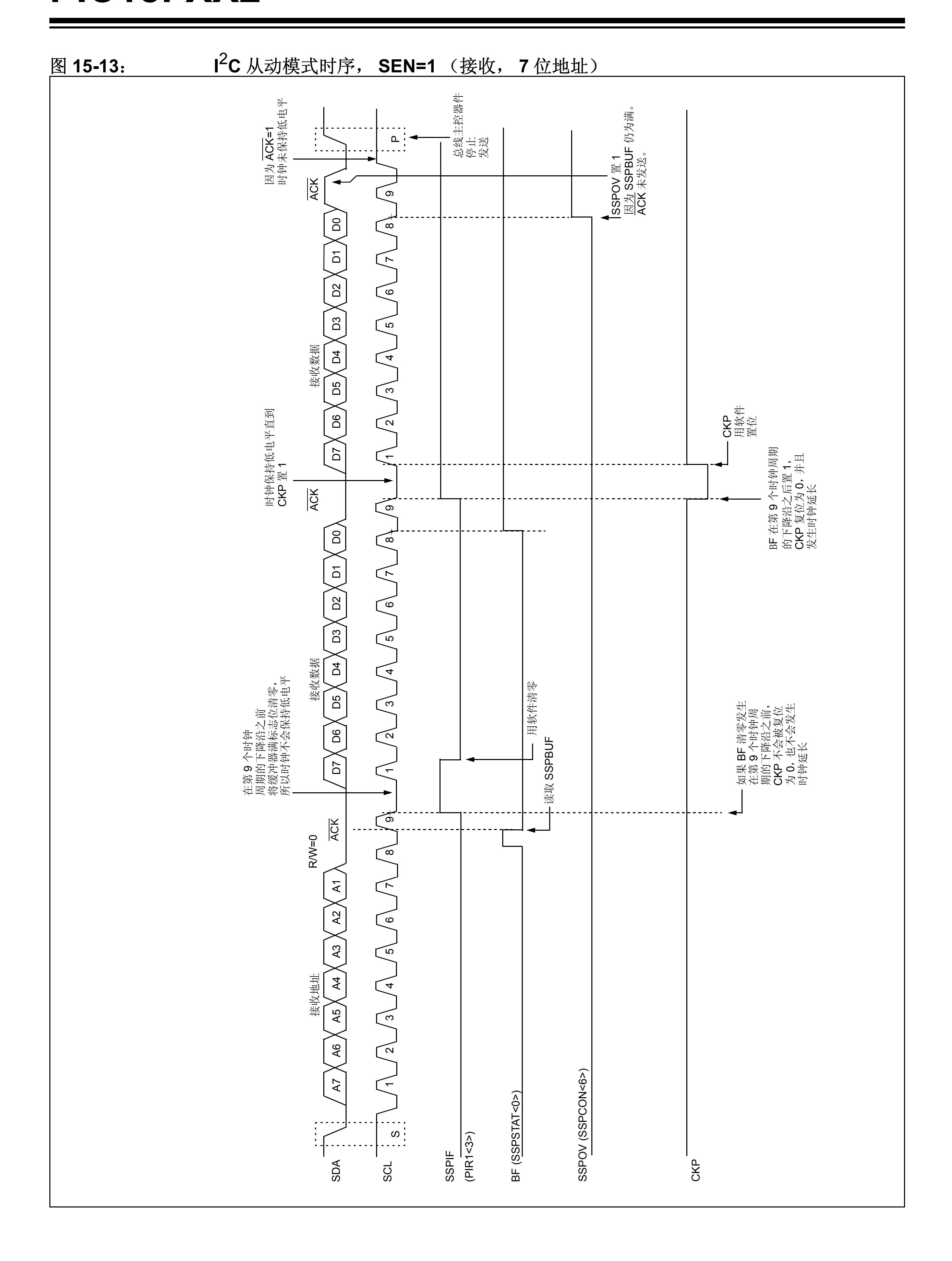
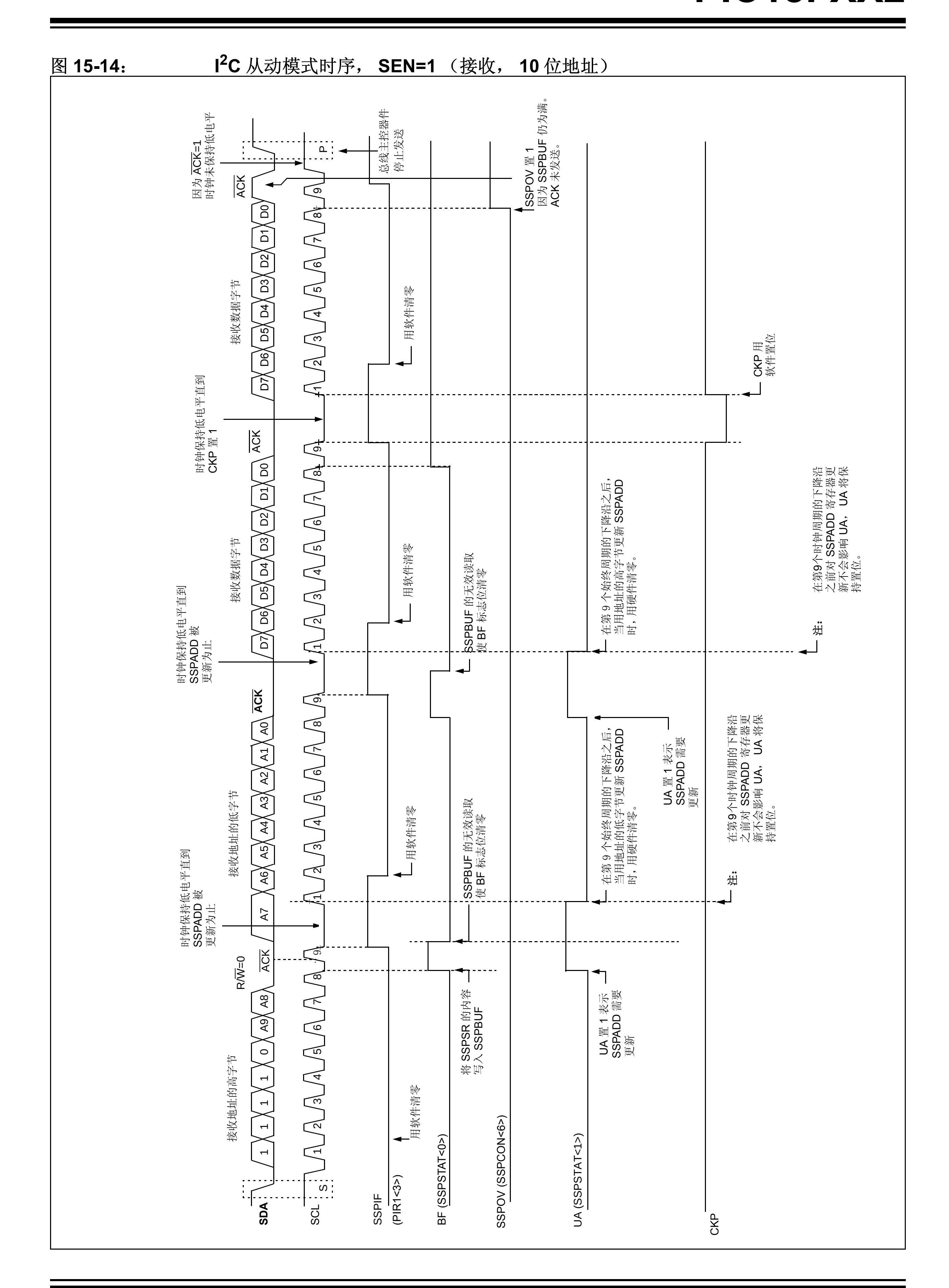


图 15-14： I²C 从动模式时序， SEN=1 （接收， 10 位地址）



15.4.5 全局呼叫地址支持

在 I²C 总线的寻址过程中，通常由启动条件后的第一个字节决定主控器件将寻址哪个从动器件。但全局呼叫地址例外，它能寻址所有器件。当使用这个地址时，理论上所有的器件都应该发送一个应答响应。

全局呼叫地址是 I²C 协议为特定用途而保留的 8 个地址之一。 R/W=0 时，此地址的所有位都是 0。

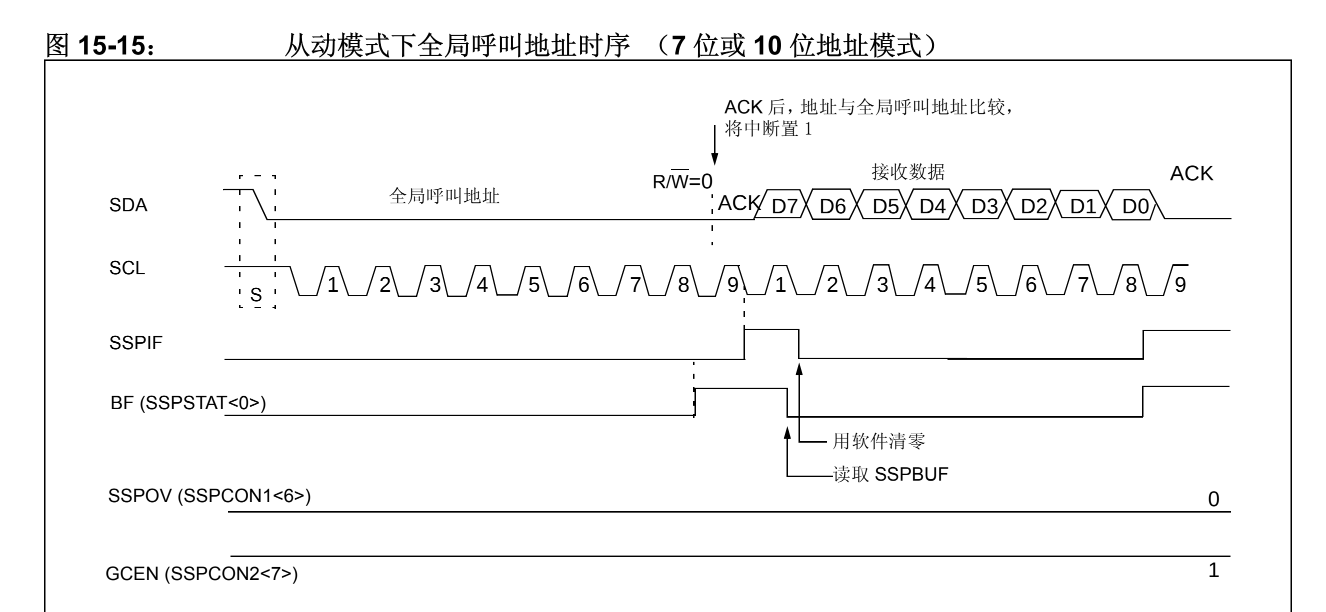
全局呼叫使能位（GCEN）使能（SSPCON2<7> 置 1）时，即可识别全局呼叫地址。检测到启动位后， 8 位数据会移入 SSPSR，并将该地址与 SSPADD 进行比较。它还会与全局呼叫地址进行比较并用硬件设定。

如果与全局呼叫地址匹配， SSPSR 的值将传递到 SSPBUF，BF 标志位（第 8 位）置 1，并且 SSPIF 中断标志位在第 9 位 （ACK 位）的下降沿置 1。

当中断得到响应时，可以通过读取 SSPBUF 的内容来检查中断源。该值可以用于判断地址是特定器件的还是一个全局呼叫地址。

在 10 位模式下，需要更新 SSPADD 使地址的后半部分匹配，同时 UA 位 （SSPSTAT<1>）置 1。如果 GCEN 位置 1 时采样到全局呼叫地址，同时从动器件被配置为 10 位地址模式，则不需要地址的后半部分，也不会将 UA 位置 1，从动器件将在应答后开始接收数据（图 15-15）。

图 15-15： 从动模式下全局呼叫地址时序 （**7** 位或 **10** 位地址模式）



15.4.6 主控模式

通过将 SSPCON1 中的相应 SSPM 位置 1 和清零，同时将 SSPEN 位置 1，可以使能主控模式。在主控模式下， SCL 和 SDA 由 MSSP 硬件控制。

主控模式通过检测启动和停止条件产生中断来工作。在复位时或禁止 MSSP 模块时，停止 （P）和启动 （S）位被清零。当 P 位被置 1，或 P 位和 S 位都清零而 I^2C 总线空闲时，可以获得对该总线的控制权。

在固件控制的主控模式中，用户代码根据启动位和停止位的状态执行所有的 I^2C 总线操作。

一旦使能主控模式，用户即可选择以下 6 项操作：

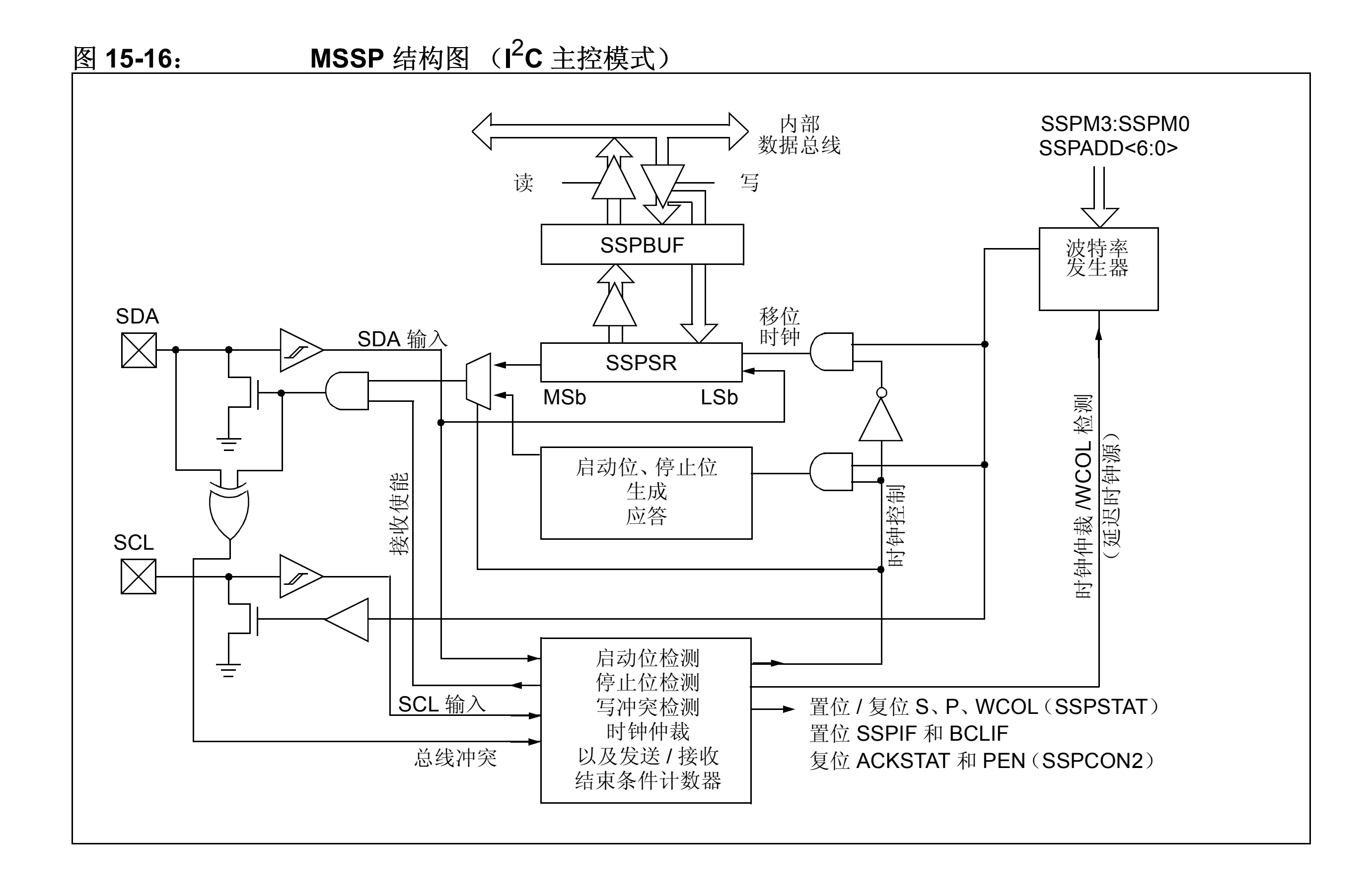
1. 在 SDA 和 SCL 上发出一个启动信号。
2. 在 SDA 和 SCL 上发出一个重复的启动信号。
3. 写入 SSPBUF 寄存器，开始数据 / 地址的发送。
4. 配置 I^2C 端口接收数据。
5. 在接收完数据字节后生成应答信号。
6. 在 SDA 和 SCL 上生成停止信号。

注：	当配置为 I^2C 主控模式时，MSSP 模块不允许事件排队。例如，在启动条件结束前，不允许用户发出启动信号并立即写 SSPBUF 寄存器来开始发送。这种情况下，将不会写入 SSPBUF， WCOL 位将被置 1，这表明没有发生对 SSPBUF 的写操作。

下列事件将导致 SSP 中断标志位 SSPIF 置 1 （如果使能 SSP 中断的话，将产生中断）：

- 启动信号
- 停止信号
- 发送 / 接收数据传输字节
- 应答发送
- 重复启动

图 15-16： MSSP 结构图 （I^2C 主控模式）



15.4.6.1 I^{2}C 主控模式工作方式

主控器件产生所有串行时钟脉冲和启动 / 停止信号。以停止信号或重复启动信号结束传送。因为重复启动信号也是下一个串行传送的开始，因此不会释放 I^{2}C 总线。

在主控发送器模式下，串行数据通过 SDA 输出，而串行时钟由 SCL 输出。发送的第一个字节包括接收器件的从动器件地址 （7 位）和读 / 写 （R/W）位。在这种情况下 R/W 位为逻辑 0。每次发送 8 位串行数据。每发送一个字节，会收到一个应答位。输出启动和停止信号分别表明串行传输的开始和结束。

在主控接收模式下，发送的第一个字节包括发送器件的从动器件地址 （7 位）和 R/W 位。在这种情况下 R/W 位为逻辑 1。因此发送的第一个字节是一个 7 位的从动器件地址，后面带有一个 1 表明接收位。串行数据通过 SDA 接收，而串行时钟由 SCL 输出。每次接收 8 位串行数据。每接收一个字节，都会发送一个应答位。启动和停止信号分别表明传输的开始和结束。

在 I^{2}C 模式下，在 SPI 模式工作中使用的波特率发生器被用于将 SCL 时钟频率设置为 100kHz、 400kHz 或 1MHz。如需更多信息，请参见第 15.4.7 节 （“波特率发生器”）。

下面是一个典型的发送序列：

1. 用户通过将启动使能位 SEN （SSPCON2<0>）置 1 来产生启动信号。
2. SSPIF 被置 1。MSSP 模块将等待所需的启动时间后，才可以执行任何其他操作。
3. 用户将从动器件地址装入 SSPBUF 进行发送。
4. 地址从 SDA 引脚移出，直到发送完所有 8 位。
5. MSSP 模块移入来自从动器件的 ACK 位，并将它的值写入 SSPCON2 寄存器 （SSPCON2<6>）。
6. MSSP 模块在第 9 个时钟周期的末尾将 SSPIF 置 1，产生一个中断。
7. 用户将 8 位数据装入 SSPBUF。
8. 数据从 SDA 引脚移出，直到发送完所有 8 位。
9. MSSP 模块移入来自从动器件的 ACK 位，并将它的值写入 SSPCON2 寄存器 （SSPCON2<6>）。
10. MSSP 模块在第 9 个时钟周期的末尾将 SSPIF 置 1，产生一个中断。
11. 用户通过将停止使能位 PEN （SSPCON2<2>）置 1 来产生停止信号。
12. 一旦停止信号完成，将产生一个中断。

15.4.7 波特率发生器

在 I²C 主控模式下，波特率发生器（baud rate generator，BRG）的重载值位于 SSPADD 寄存器的低 7 位（图 15-17）。当发生对 SSPBUF 的写操作时，波特率发生器自动开始计数。BRG 会递减计数至 0，然后停止，并等待另一次的重新载入。BRG 计数会在每个指令周期（TCY）中的 Q2 和 Q4 时钟周期上分别递减。在 I²C 主控模式下，会自动重新装入 BRG。

一旦完成了指定操作（即，在发送的最后数据位后面带有 ACK），内部时钟将自动停止计数，SCL 引脚将保持在其最后的状态。

表 15-3 显示了不同指令周期下的时钟频率以及装入 SSPADD 的 BRG 值。

图 15-17： 波特率发生器结构图

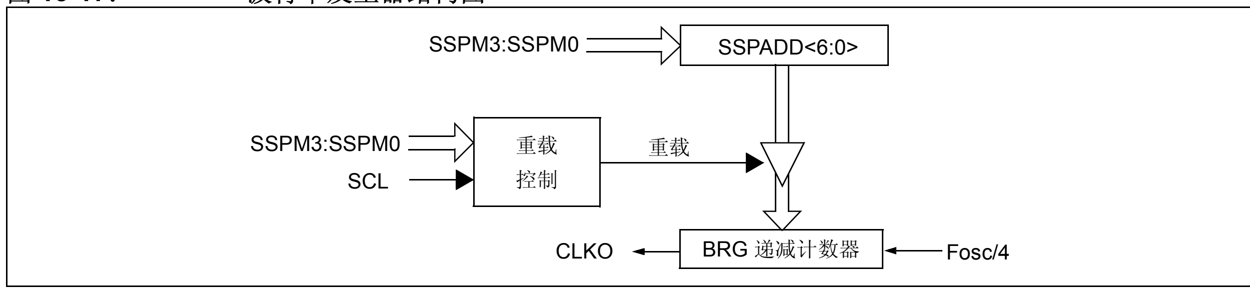


表 15-3： **I²C 时钟频率与 BRG**

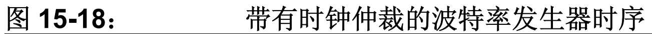
FCY	FCY*2	BRG 值	FSCL(2)（两次 BRG 翻转）
10 MHz	20 MHz	19h	400 kHz(1)
10 MHz	20 MHz	20h	312.5 kHz
10 MHz	20 MHz	3Fh	100 kHz
4 MHz	8 MHz	0Ah	400 kHz(1)
4 MHz	8 MHz	0Dh	308 kHz
4 MHz	8 MHz	28h	100 kHz
1 MHz	2 MHz	03h	333 kHz(1)
1 MHz	2 MHz	0Ah	100kHz
1 MHz	2 MHz	00h	1 MHz(1)

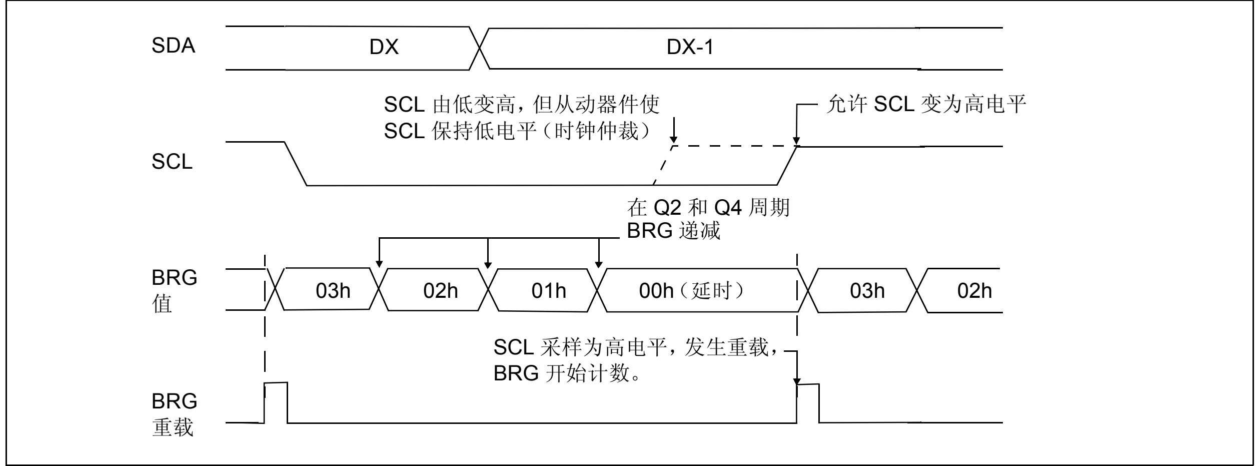
注 **1：** 虽然 I²C 接口各方面都不符合 400 kHz I²C 规范（该规范适用于大于 100 kHz 的频率），但在需要较高频率的应用场合可以小心使用。

2： 实际频率将因总线状态而异。理论上，总线状态会增加上升时间并延长时钟周期的低电平时间，从而产生有效频率。

15.4.7.1 时钟仲裁

如果在任何接收、发送或重复启动 / 停止信号中，主控器件拉高了 SCL 引脚（允许 SCL 引脚悬空为高电平），就会发生时钟仲裁。如果允许 SCL 引脚悬空为高电平，波特率发生器（BRG）将暂停计数直到实际采样到 SCL 引脚为高电平。当 SCL 引脚采样为高电平时，波特率发生器将被重新装入 SSPADD<6:0> 的内容并开始计数。这可以保证即使外部器件将时钟拉低时，SCL 始终至少保持一个 BRG 翻转计数周期的高电平（图 15-18）。


图 **15-18：** 带有时钟仲裁的波特率发生器时序



15.4.8 I²C 主控模式启动条件时序

为开始启动条件，用户应将启动条件使能位 SEN（SSPCON2<0>）置 1。当 SDA 和 SCL 引脚都采样为高电平时，波特率发生器重新装入 SSPADD<6:0> 的内容并开始计数。如果波特率发生器发生超时（TBRG）时 SCL 和 SDA 都采样为高电平，则 SDA 引脚被驱动为低电平。当 SCL 为高电平时，将 SDA 由高电平拉到低电平就是启动条件，并使 S 位（SSPSTAT<3>）置 1。随后波特率发生器重新装入 SSPADD<6:0> 的内容并重新开始计数。当波特率发生器超时（TBRG）时，SEN 位（SSPCON2<0>）将自动被硬件清零，波特率发生器暂停工作，SDA 保持低电平，启动条件结束。

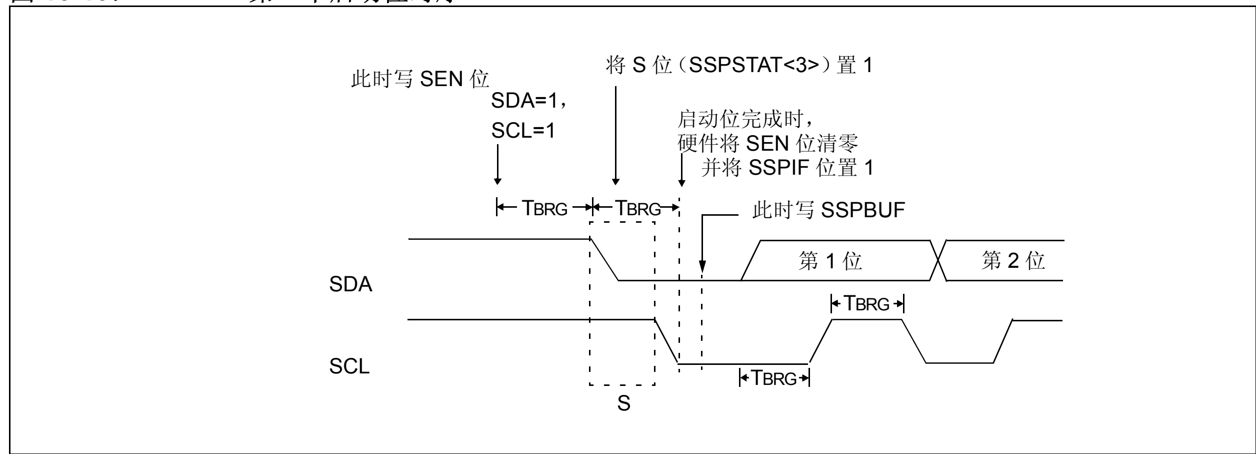
注：	如果在启动条件开始时，SDA 和 SCL 引脚已经采样为低电平，或者启动条件中 SCL 在 SDA 被拉低之前已经采样为低电平，则会产生总线冲突，总线冲突标志位 BCLIF 置 1，启动条件中止，I²C 模块复位到空闲状态。

15.4.8.1 WCOL 状态标志位

当启动序列进行时，如果用户试图写 SSPBUF，则 WCOL 将置 1，缓冲器的内容不变（写操作无效）。

注：	由于不允许事件排队，在启动条件结束之前，禁止对 SSPCON2 的低 5 位进行写操作。

图 15-19： 第一个启动位时序



15.4.9 I²C 主控模式重复启动条件时序

将 RSEN 位（SSPCON2<1>）编程为高电平，并且 I²C 逻辑模块处于空闲状态时，就会产生重复启动条件。当 RSEN 位置 1 时， SCL 引脚为低电平有效。当 SCL 引脚采样为低电平时，波特率发生器装入 SSPADD<5:0> 的内容，并开始计数。在一个波特率发生器计数周期（TBRG）内 SDA 引脚被释放（其引脚电平被拉高）。当波特率发生器超时时，如果 SDA 采样为高电平，SCL 引脚将被拉高。当 SCL 被采样为高电平时，波特率发生器重新装入 SSPADD<6:0> 的内容并开始计数。在一个计数周期 TBRG 内，SDA 和 SCL 必须保持为高电平。接下来，当 SCL 为高电平时，在下一个 TBRG 中，将 SDA 引脚拉为低电平。然后 RSEN 位（SSPCON2<1>）将自动清零，波特率发生器不再重新装入值，SDA 引脚保持低电平。一旦在 SDA 和 SCL 引脚上检测到启动条件，S 位（SSPSTAT<3>）将置 1。直到波特率发生器超时后， SSPIF 位才会置 1。

> 注 **1**：有任何其他事件在进行时，编程对 RSEN 进行设置无效。
>
> **2**：在重复启动条件期间，下列事件将会导致总线冲突：
> - 当 SCL 由低电平变为高电平时，SDA 采样为低电平。
> - SDA 被拉低之前， SCL 变为低电平。这可能表明另一个主控器件正试图发送一个数据 “1”。

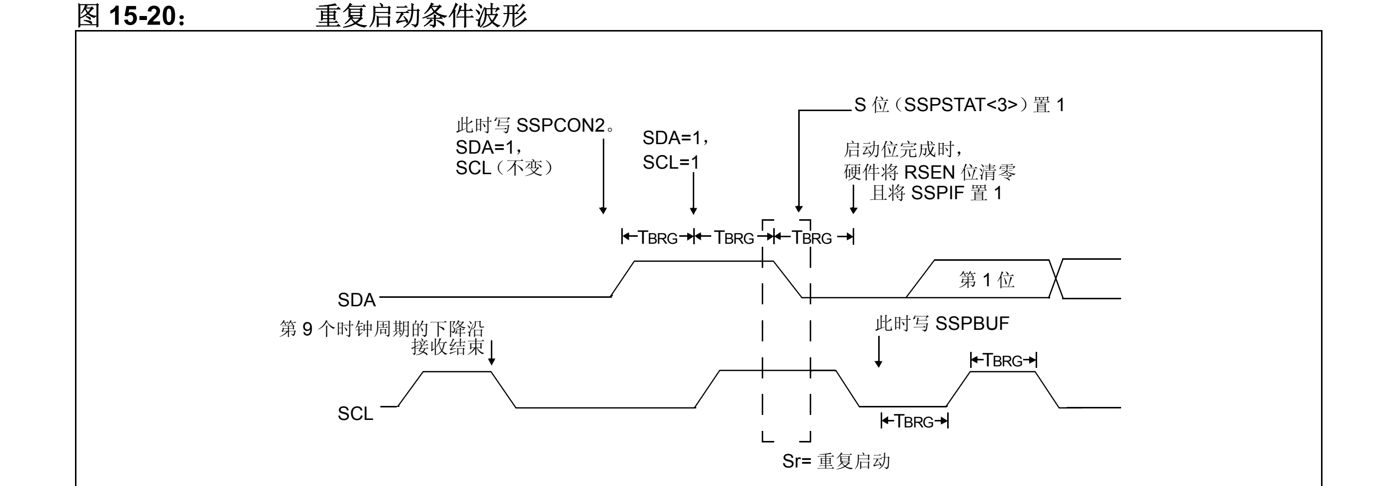
一旦 SSPIF 位置 1，用户便可以在 7 位地址模式下将 7 位地址写入 SSPBUF，或者在 10 位地址模式下写入默认的第一个地址字节。当发送完第一个 8 位数据并接收到一个 ACK 后，用户可以发送另一个 8 位地址（10 位地址模式下）或 8 位数据（7 位地址模式下）。

15.4.9.1 WCOL 状态标志位

当重复启动序列进行时，如果用户试图写 SSPBUF，则 WCOL 将置 1，同时缓冲器内容不变（写操作无效）。

> **注：** 由于不允许事件排队，在重复启动条件结束之前，禁止对 SSPCON2 的低 5 位进行写操作。

图 **15-20**： 重复启动条件波形



15.4.10 I^2C 主控模式下的发送

发送一个数据字节、一个 7 位地址或一个 10 位地址的另一半，都可以通过写一个值到 SSPBUF 寄存器来实现。该操作将使缓冲器满标志位 BF 置 1，并且波特率发生器开始计数，同时启动下一次发送。在 SCL 的下降沿有效后 （见数据保持时间规范参数 106），地址 / 数据的每一位都从 SDA 引脚移出。在一个波特率发生器翻转计数周期（TBRG）内， SCL 保持低电平。数据应该在 SCL 释放为高电平前保持有效 （见数据建立时间规范参数 107）。当 SCL 引脚释放为高电平时，它将在 TBRG 内保持高电平状态。在此期间以及下一个 SCL 下降沿之后的一段时间内，SDA 引脚上的数据必须保持稳定。在 8 位都移出之后 （第 8 个时钟周期的下降沿），BF 标志位清零，同时主控器件释放 SDA。此时如果发生地址匹配或是数据正确接收，所寻址的从动器件将在第 9 位时以一个 ACK 位响应。ACK 的状态在第 9 个时钟周期的下降沿写入 ACKDT 位。主控器件收到应答响应之后，应答状态位 ACKSTAT 会被清零。如果未收到响应，则该位被置 1。第 9 个时钟周期之后， SSPIF 位会置 1，主时钟 （波特率发生器）暂停，直到下一个数据字节装入 SSPBUF， SCL 引脚保持低电平， SDA 保持不变 （图 15-21）。

在写 SSPBUF 之后，地址的每一位在 SCL 的下降沿被移出，直至所有 7 位地址和 R/$\overline{\text{W}}$ 位都移出。在第 8 个时钟下降沿，主控器件将 SDA 引脚上拉为高电平，以允许从动器件发出一个应答响应。在第 9 个时钟下降沿，主控器件通过采样 SDA 引脚来判断地址是否被从动器件识别。 ACK 位状态被装入 ACKSTAT 状态位（SSPCON2<6>）。在地址发送的第 9 个时钟下降沿之后，SSPIF 置 1，BF 标志位清零，波特率发生器关闭直到下一次写 SSPBUF，且 SCL 引脚保持低电平，允许 SDA 引脚悬空。

15.4.10.1 BF 状态标志位

在发送模式下， BF 位 （SSPSTAT<0>）在 CPU 写 SSPBUF 时置 1，在所有 8 位数据移出后清零。

15.4.10.2 WCOL 状态标志位

如果用户在发送过程中（即 SSPSR 仍在移出数据字节时）试图写 SSPBUF，则 WCOL 置 1，缓冲器内容不变 （写操作无效）。

WCOL 必须用软件清零。

15.4.10.3 ACKSTAT 状态标志位

在发送模式下，当从动器件发送应答响应（$\overline{\text{ACK}}$=0）时，ACKSTAT 位（SSPCON2<6>）清零，当从动器件没有应答（ACK=1）时，该位置 1。从动器件在识别出其地址（包括全局呼叫地址）或正确接收数据后，会发送一个应答信号。

15.4.11 I^2C 主控模式接收

通过编程接收使能位 RCEN （SSPCON2<3>）为 1，进入主控模式接收。

注：	在 MSSP 模块中，必须在 ACK 序列后才能将 RCEN 位置 1，否则将忽略 RCEN 位的状态。

波特率发生器开始计数，每次翻转时，SCL 引脚的状态发生改变（由高变低或由低变高），数据被移入 SSPSR。第 8 个时钟下降沿之后，接收使能标志位自动清零，SSPSR 的内容装入 SSPBUF，BF 标志位置 1，SSPIF 标志位置 1，波特率发生器暂停计数，SCL 保持为低电平。此时 MSSP 进入空闲状态，等待下一条命令。当 CPU 读缓冲器时， BF 标志位将自动清零。然后，通过将应答序列使能位 ACKEN（SSPCON2<4>）置 1，用户可以在接收结束后发出应答信号。

15.4.11.1 BF 状态标志位

接收时，当将地址或数据字节从 SSPSR 装入 SSPBUF 时，BF 位置 1。在读 SSPBUF 寄存器时将 BF 位清零。

15.4.11.2 SSPOV 状态标志位

接收时，如果在 SSPSR 接收到 8 位数据时，BF 标志位已经在上一次接收时置 1，则 SSPOV 位置 1。

15.4.11.3 WCOL 状态标志位

如果用户在接收过程中（即 SSPSR 仍在移入数据字节时）试图写 SSPBUF，则 WCOL 置 1，缓冲器内容不变 （写操作无效）。

图 15-21： I²C 主控模式时序 （发送， 7 位或 10 位地址）

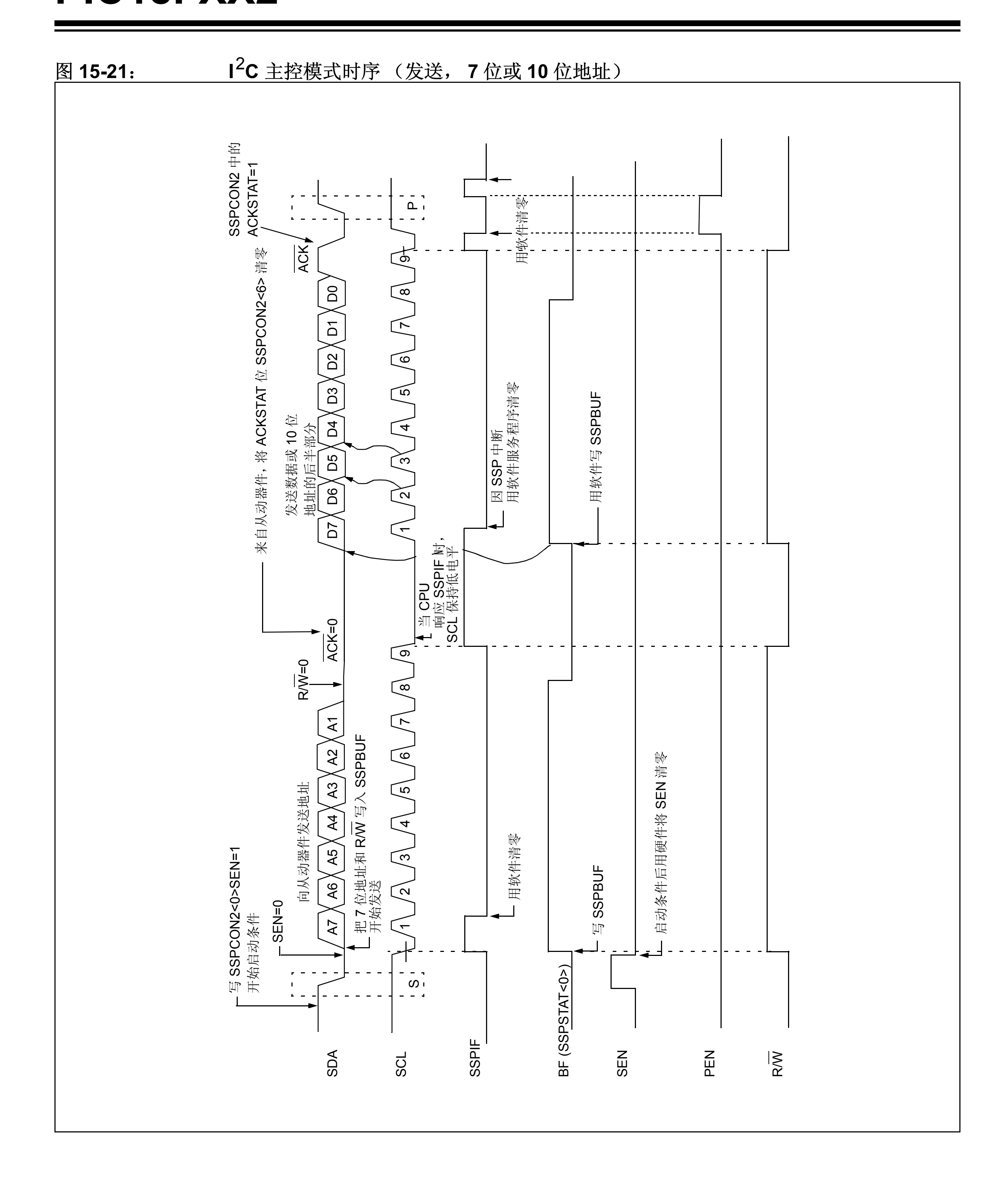
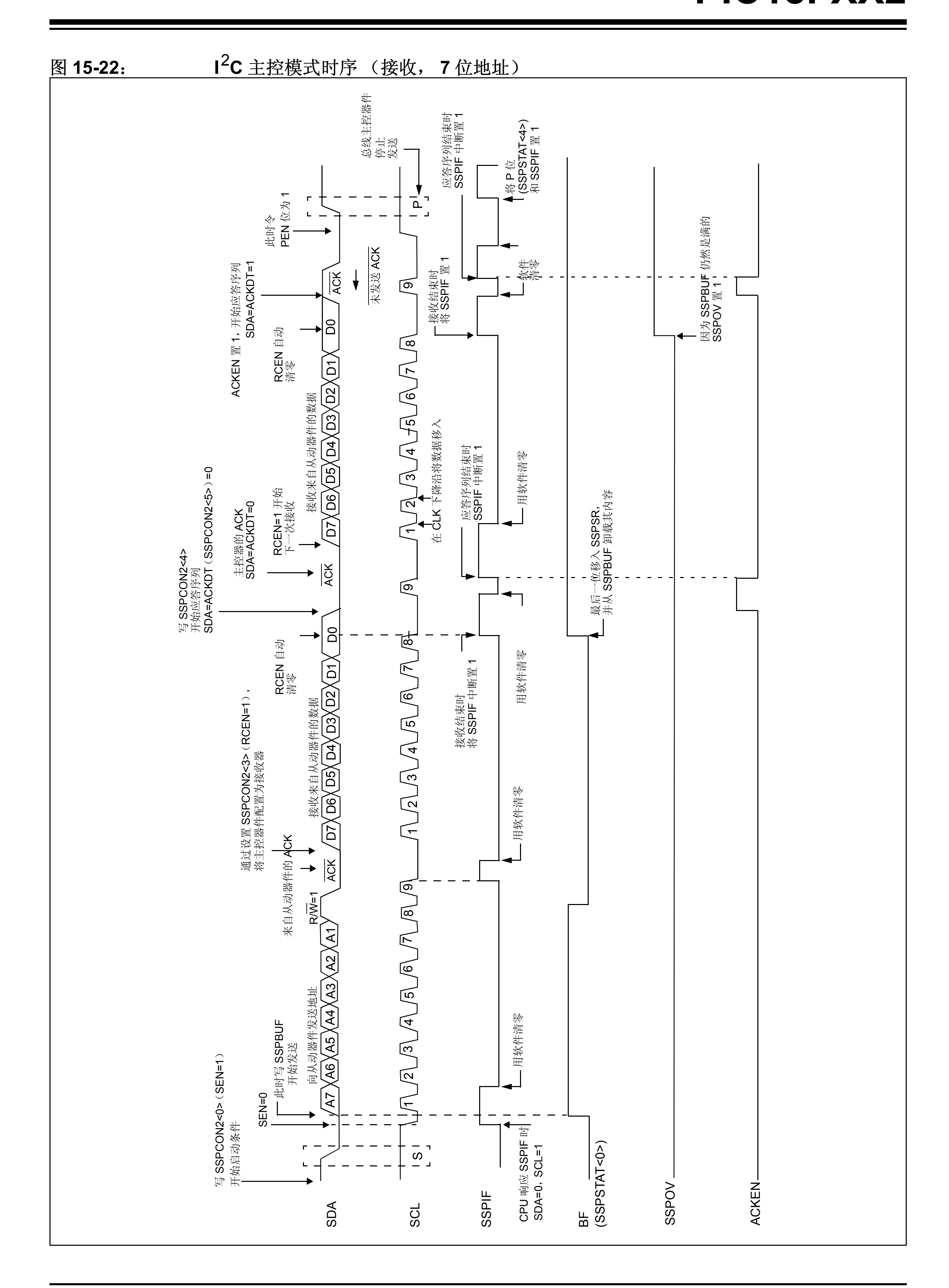


图 15-22：　　I²C 主控模式时序 （接收， 7 位地址）



15.4.12 应答序列时序

将应答序列使能位 ACKEN（SSPCON2<4>）置 1 即可使能应答序列。当该位被置 1 时，拉低 SCL 引脚，应答数据位的内容出现在 SDA 引脚上。如果用户希望产生一个应答，应将 ACKDT 位清零。不然，用户当在应答序列开始前将 ACKDT 位置 1。波特率发生器进行一个翻转周期（TBRG）的计数，接着 SCL 引脚被释放（被拉高）。当 SCL 引脚采样为高电平（时钟仲裁）时，波特率发生器进行一个 TBRG 周期的计数。然后 SCL 引脚被拉低。之后，ACKEN 位自动清零，波特率发生器关闭，MSSP 模块进入空闲模式（图 15-23）。

15.4.12.1 WCOL 状态标志位

如果用户在应答序列进行中试图写 SSPBUF，则 WCOL 将置 1，缓冲器的内容不变（写操作无效）。

15.4.13 停止条件时序

将停止序列使能位 PEN（SSPCON2<2>）置 1，在接收 / 发送结束后，SDA 引脚上产生低电平的停止位。在接收 / 发送结束后，SCL 在第 9 个时钟下降沿之后保持低电平。当 PEN 位置 1 时，主控器件将 SDA 置为低电平。当 SDA 采样为低电平时，将重新装入波特率发生器并递减计数至 0。当波特率发生器超时时，SCL 引脚被拉到高电平，在一个 TBRG（波特率发生器翻转周期）后，SDA 引脚将被拉高。当 SDA 引脚采样为高电平且 SCL 也是高电平时，P 位（SSPSTAT<4>）置 1。一个 TBRG 之后，PEN 位清零，同时 SSPIF 位置 1（图 15-24）。

15.4.13.1 WCOL 状态标志位

如果用户在停止序列进行中试图写 SSPBUF，则 WCOL 位将置 1，缓冲器的内容不变（写操作无效）。

图 15-23： 应答序列时序

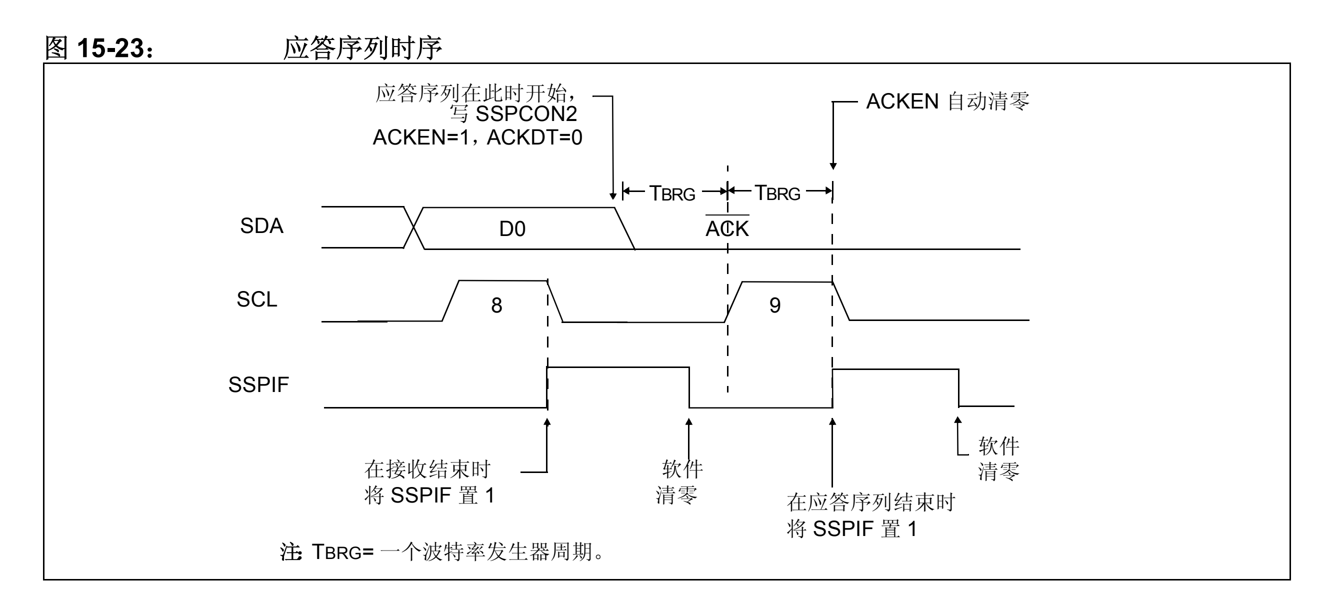
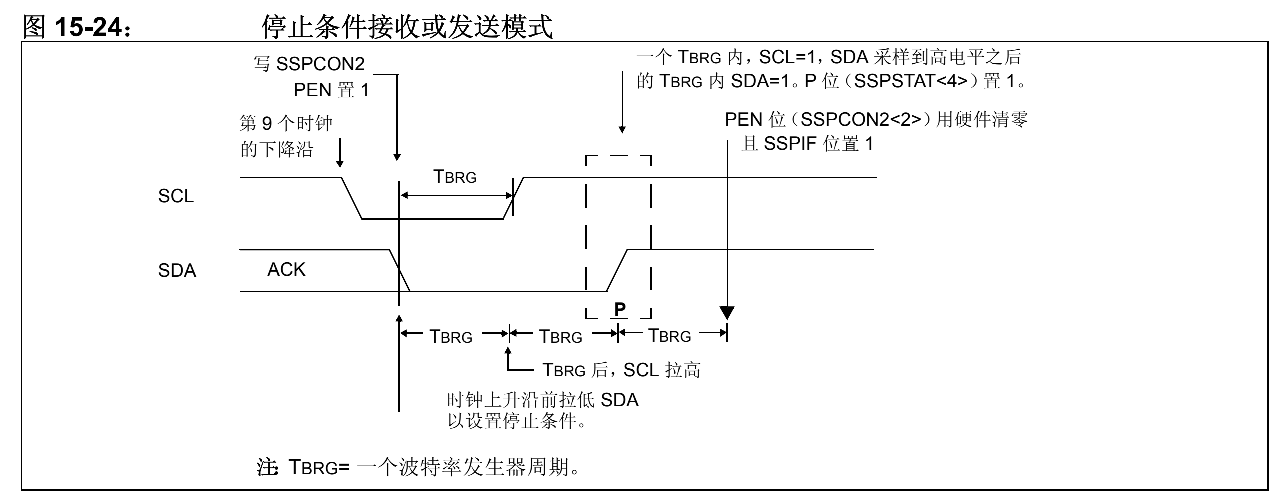


图 15-24： 停止条件接收或发送模式



15.4.14 休眠操作

在休眠模式下，I^2C 模块能够接收地址或数据。并且当地址匹配或字节传输完成后，如果使能 MSSP 中断，则将处理器从休眠状态唤醒。

15.4.15 复位的影响

复位可以禁用 MSSP 模块并终止当前传输。

15.4.16 多主控模式

在多主控模式下，通过检测启动条件和停止条件产生中断，可以确定总线何时空闲。在复位时或禁止 MSSP 模块时，停止位 （P）和启动位 （S）被清零。当 P 位（SSPSTAT<4>）置 1，或 P 位和 S 位都为零而总线空闲时，可以取得I^2C总线的控制权。当总线忙时使能SSP中断，将在发生停止条件时产生中断。

在多主控模式中，必须监视 SDA 来进行仲裁，查看信号电平是是否期望的输出电平。此检查在硬件中进行，其结果放在 BCLIF 位。

下列情况中可能会丢失对总线的仲裁：

- 地址传输
- 数据传输
- 启动条件
- 重复启动条件
- 应答条件

15.4.17 多主控通讯、总线冲突与总线仲裁

多主控模式支持是通过总线仲裁来实现的。当主控器件将地址 / 数据位输出到 SDA 引脚时，如果主控器件通过将 SDA 引脚悬空为高电平来在 SDA 上输出 1，而另一个主控器件输出 0，就会发生总线仲裁。当 SCL 引脚悬空为高电平时，数据应保持稳定。如果 SDA 引脚上期望的数据是 1，而实际采样到的数据是 0，则发生了总线冲突。主控器件将把总线冲突中断标志位 （BCLIF）置 1，并将 I^2C 端口复位到空闲状态 （图 15-25）。

如果在发送过程中发生总线冲突，则中止发送， BF 标志位清零，SDA 和 SCL 拉高，并且 SSPBUF 可写。当执行完总线冲突中断服务程序后，如果 I^2C 总线空闲，用户可通过发出启动条件恢复通讯。

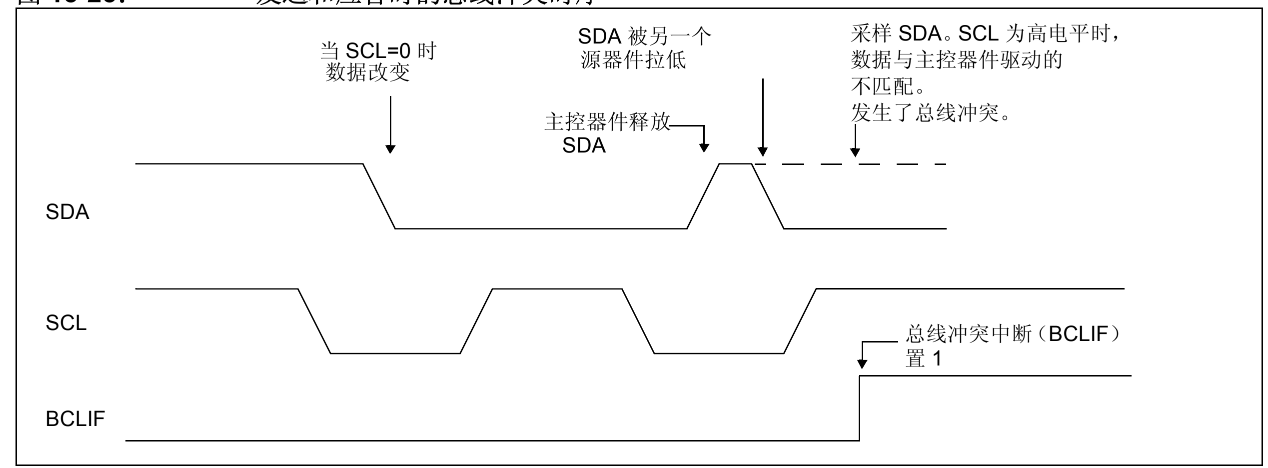
如果在启动、重复启动、停止或应答条件的执行过程中发生总线冲突，则这种条件被中止，SDA 和 SCL 拉高，SSPCON2 寄存器中的相应控制位清零。当执行完总线冲突中断服务程序后，如果 I^2C 总线空闲，用户可通过发出启动条件恢复通讯。

主控器件将继续监控 SDA 和 SCL 引脚。如果出现停止条件， SSPIF 位将被置 1。

无论发生总线冲突时发送的进度如何，写 SSPBUF 都会从第一个数据位开始发送数据。

在多主控模式下，可通过检测启动和停止条件产生中断，以确定总线何时空闲。 SSPSTAT 寄存器中的 P 位置 1，或总线空闲且 S 和 P 位清零时，可以获取 I^2C 总线的控制权。

图 15-25： 发送和应答时的总线冲突时序



15.4.17.1 启动条件中的总线冲突

在启动条件下，以下事件将导致总线冲突：

a） 在启动条件开始时， SDA 或 SCL 采样为低电平（图 15-26）。

b） SDA 被拉低之前， SCL 采样为低电平（图 15-27）。

在启动条件中，会监控 SDA 和 SCL 引脚。

如果 SDA 引脚已经是低电平，或 SCL 引脚已经是低电平，则：

- 中止启动条件，
- BCLIF 标志置 1，
- MSSP 模块复位为空闲状态 （图 15-26）。

启动条件从 SDA 和 SCL 引脚被拉高开始。当 SDA 引脚采样为高电平时，波特率发生器装入 SSPADD<6:0> 的内容并递减计数至 0。如果在 SDA 为高电平时，SCL 引脚采样为低电平，则发生总线冲突，因为这表示另一个主控器件在启动条件期间试图发送一个数据 “1”。

如果SDA引脚在这一计数期间内采样为低电平，则BRG 复位，同时 SDA 线保持原值（图 15-28）。但是，如果 SDA 引脚采样为 1，SDA 引脚将在 BRG 计数结束时被置为低电平。接着，波特率发生器被重新装入值并递减计数至 0，在此期间，如果 SCL 引脚采样为 0，则不会发生总线冲突。在 BRG 计数结束时， SCL 引脚被置为低电平。

注：	在启动条件下不会发生总线冲突是因为总线上的两个主控器件不可能精确地在同一时刻发出启动信号，因此一个主控器件总是先于另一个主控器件将 SDA 拉低。但是这一情况不会引起总线冲突，因为必须允许两个主控器件对启动条件后的第一个地址进行仲裁。如果是同一地址，必须继续对数据部分、重复启动条件或停止条件进行仲裁。

图 15-26： 启动条件下的总线冲突 （仅 SDA）

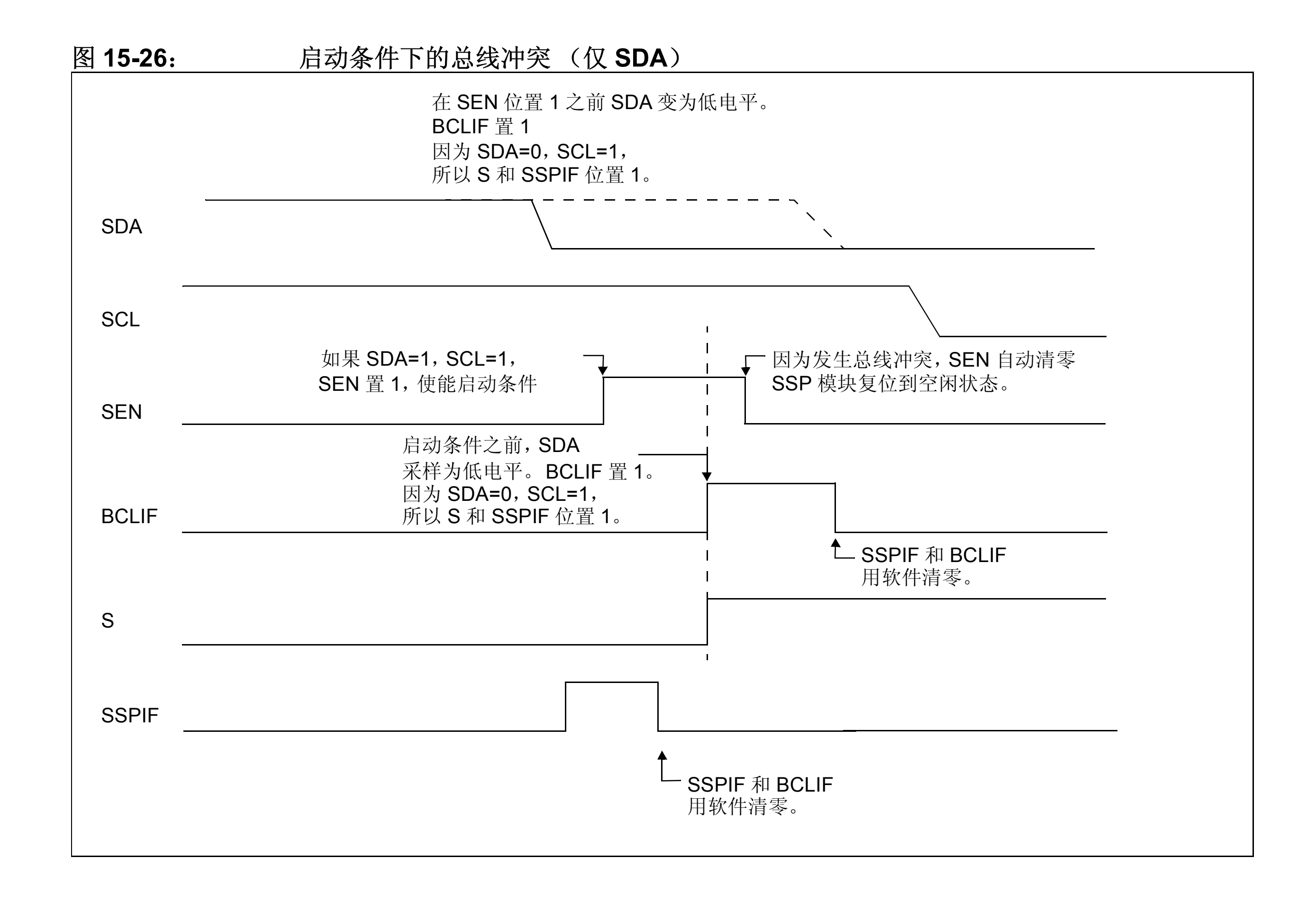


图 15-27： 启动条件下的总线冲突 （**SCL=0**）

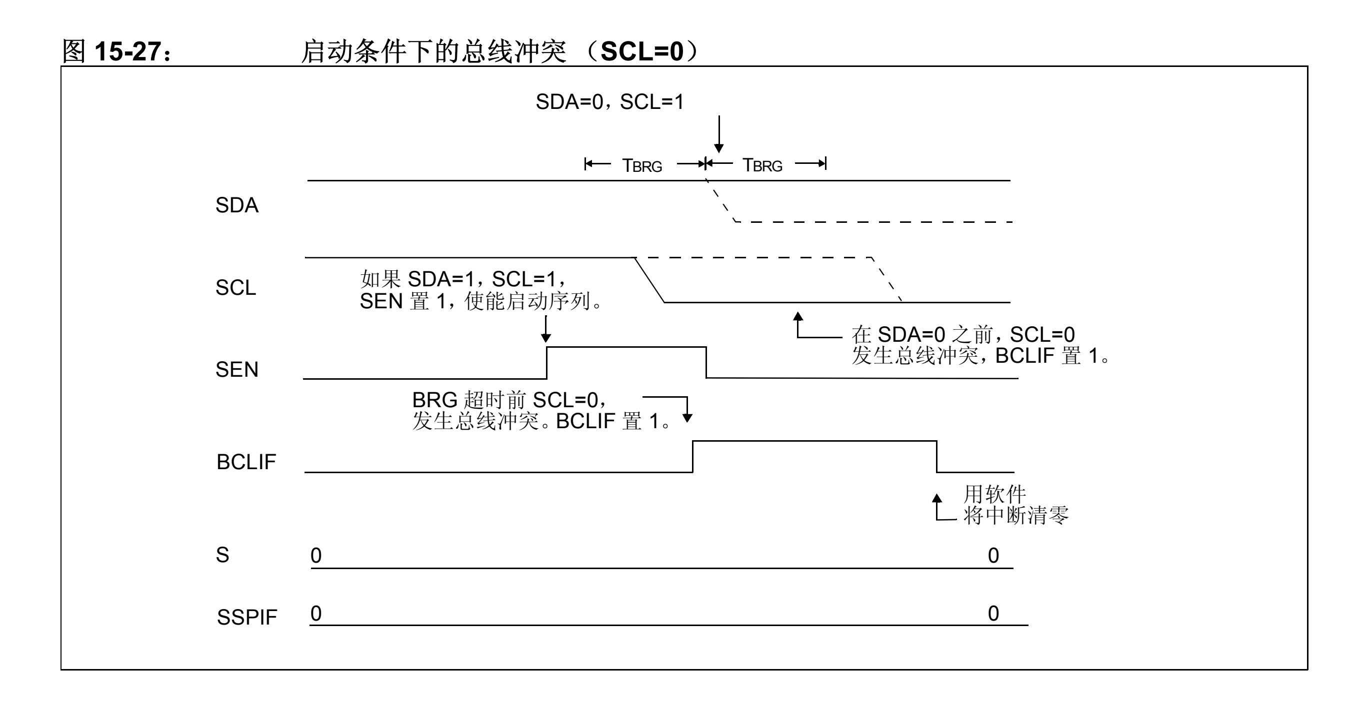
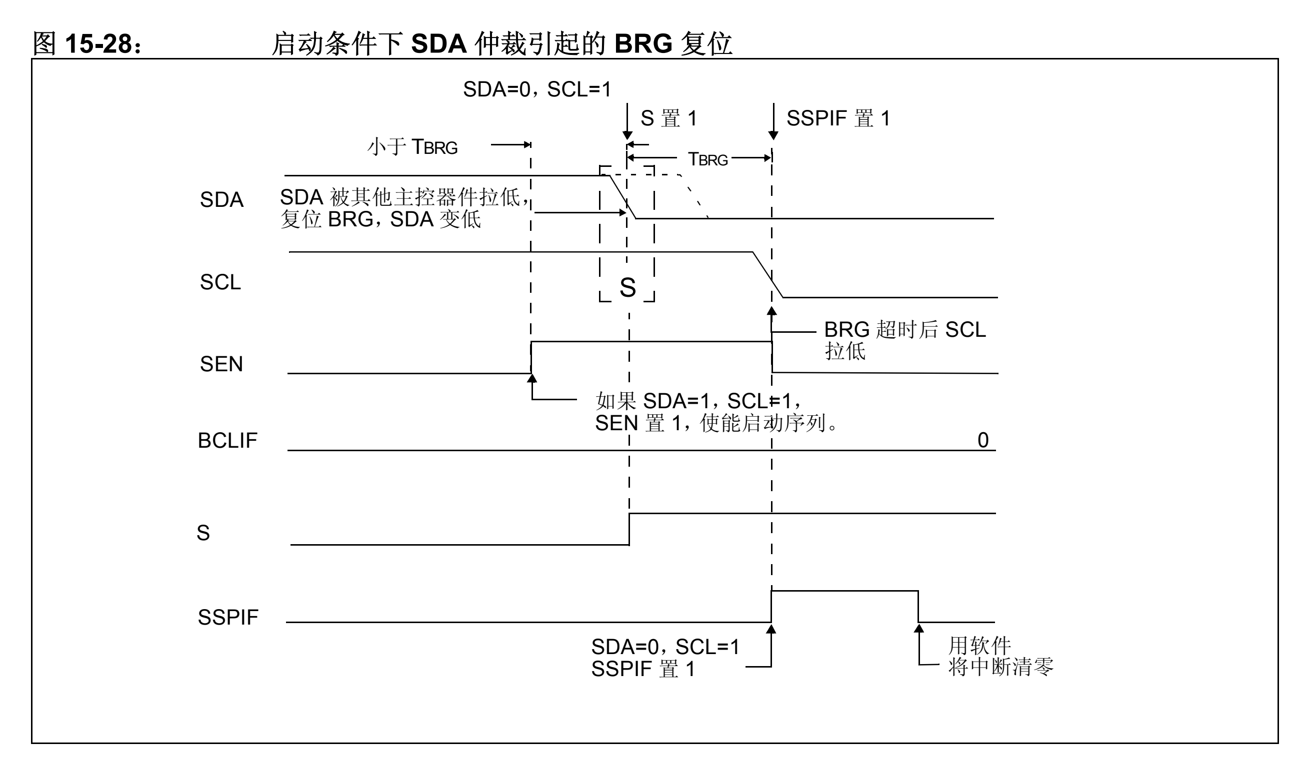


图 15-28： 启动条件下 **SDA** 仲裁引起的 **BRG** 复位



15.4.17.2 重复启动条件下的总线冲突

在重复启动条件下，以下事件将导致总线冲突：

a） 在 SCL 由低电平变为高电平的过程中，在 SDA 上采样到低电平。

b） 在 SDA 被置为低电平之前， SCL 变为低电平，表示另一个主控器件正试图发送一个数据 “1”。

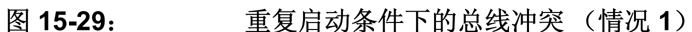
当用户拉高 SDA 且允许该引脚悬空为高电平时， BRG 装入 SSPADD<6:0> 中的值并递减计数至零。接着 SCL 引脚被拉高，当 SCL 采样为高电平时，对 SDA 引脚进行采样。

如果 SDA 为低电平，则发生总线冲突 （即另一个主控器件正试图发送一个数据 “0”， 如图 15-29）。如果 SDA采样为高电平，则BRG被重新装入值并开始计数。

如果 SDA 在 BRG 超时之前从高电平变为低电平，则不会发生总线冲突，因为两个主控器件不可能精确地在同一时刻将 SDA 拉低。

如果 SCL 在 BRG 超时之前从高电平变为低电平，且 SDA 尚未变为低电平，那么将发生总线冲突。这种情况表示另一个主控器件在重复启动条件过程中正试图发送一个数据 “1”，如图 15-30。

如果在BRG超时结束时SCL和SDA都仍然是高电平，则 SDA 引脚被拉低， BRG 重新装入值并开始计数。在计数结束时，不管 SCL 引脚状态如何， SCL 引脚都被拉低，重复启动条件结束。


图 15-29： 重复启动条件下的总线冲突 （情况 1）

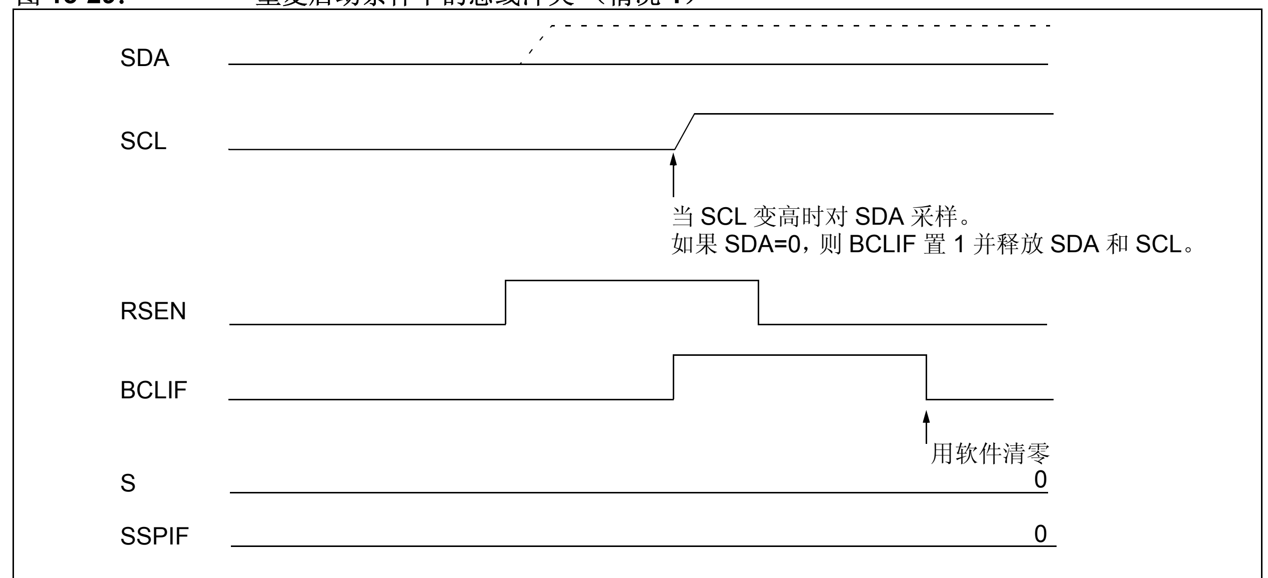
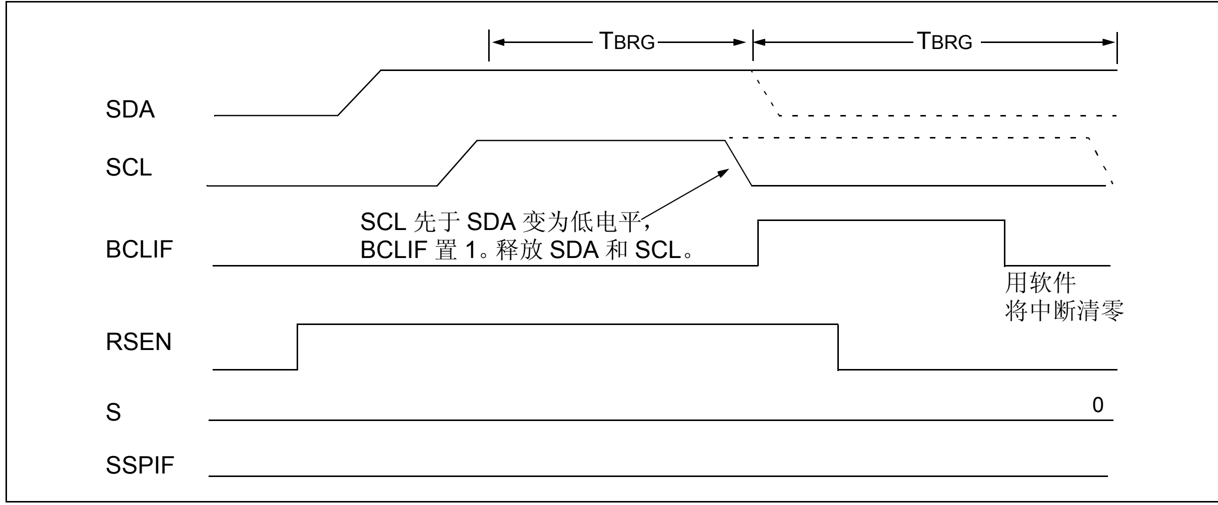


图 15-30： 重复启动条件下的总线冲突 （情况 2）



15.4.17.3 停止条件下的总线冲突

在停止条件下，以下事件将导致总线冲突：

a） SDA引脚被拉高并允许悬空为高电平之后，SDA 在 BRG 超时后被采样到低电平。

b） SCL 引脚被拉高之后，SCL 在 SDA 变成高电平之前被采样到低电平。

停止条件从 SDA 被拉低开始。 SDA 采样为低电平时，SCL 引脚就可以悬空。当引脚被采样为高电平（时钟仲裁）时，波特率发生器装入 SSPADD<6:0> 的值并递减计数到 0。在 BRG 超时后，对 SDA 采样。如果 SDA 采样为低电平，则已发生总线冲突。这是因为另一个主控器件正试图发送一个数据 “0”（如图 15-31）。如果 SCL 引脚在允许 SDA 悬空为高电平前采样为低电平，也会发生总线冲突。这是另一个主控器件正试图发送一个数据 “0” 的另一种情况 （如图 15-32）。

图 15-31： 停止条件下的总线冲突 （情况 **1**）

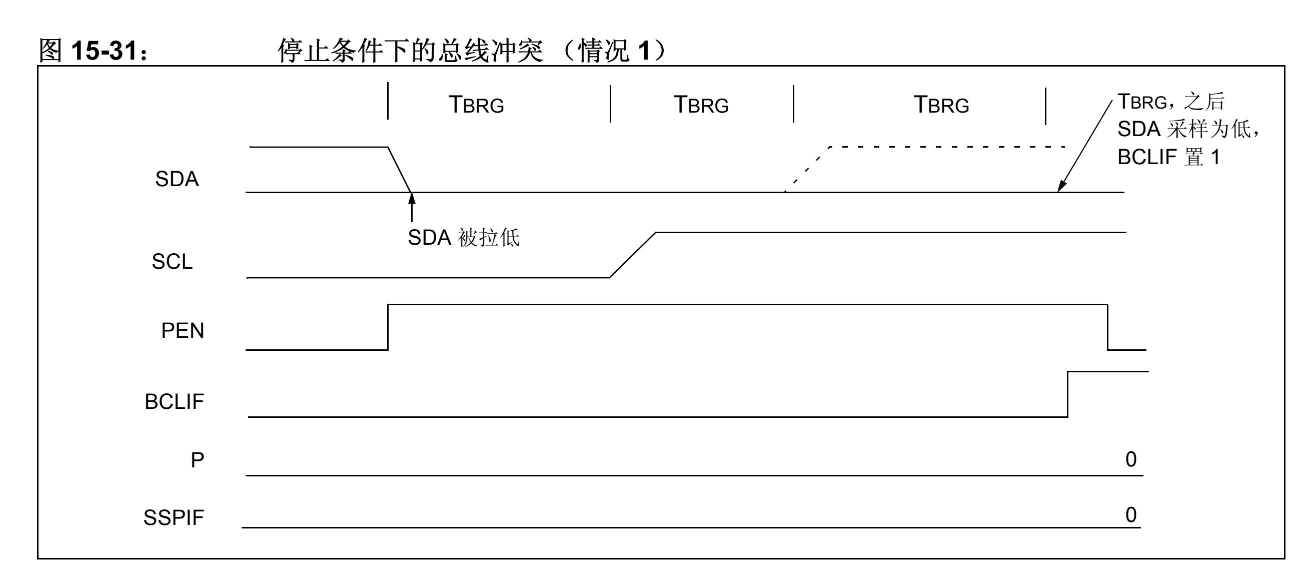
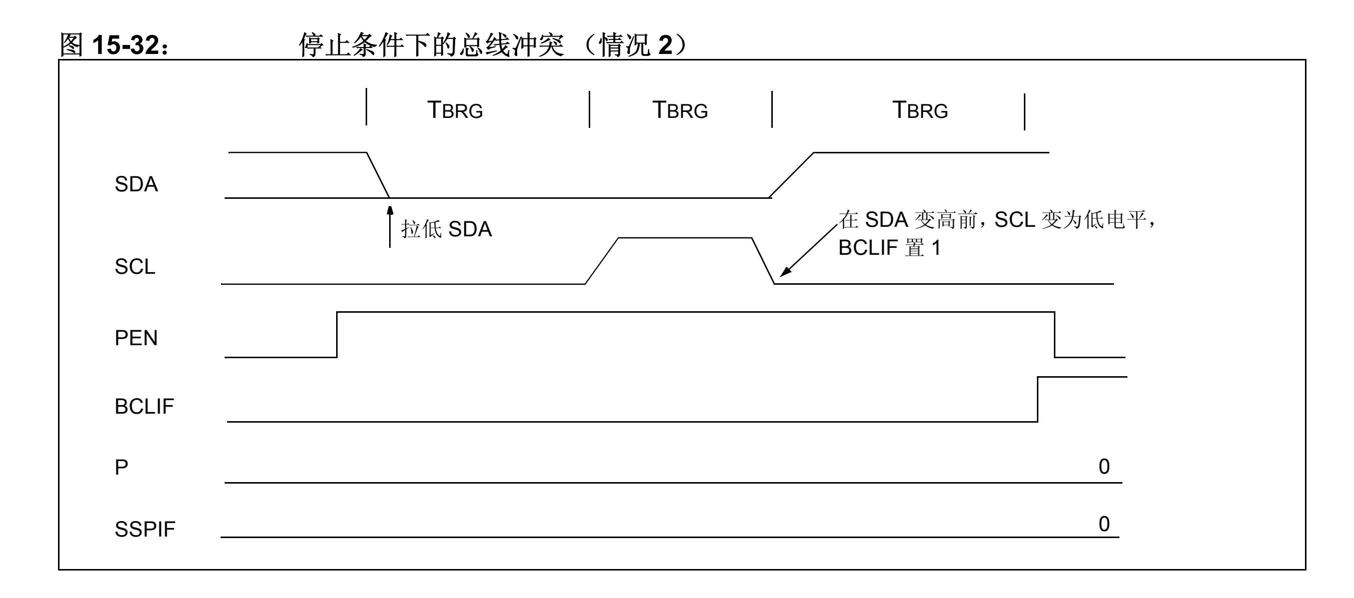


图 15-32： 停止条件下的总线冲突 （情况 **2**）



注：

16.0 可寻址的通用同步 / 异步收发器 （USART）

通用同步 / 异步收发器（Universal Synchronous Asynchronous receiver Transmitter， USART）模块是两个串行 I/O 模块之一。（USART 也称为串行通讯接口（Serial Communications Interface， SCI）。）USART 可以配置为全双工异步系统，实现与 CRT 终端和个人计算机等外围设备进行通讯；也可配置为半双工同步系统，实现与 A/D 或 D/A 集成电路及串行 EEPROM 等外围器件进行通讯。

USART 可配置为以下几种工作模式：

- 全双工异步模式
- 半双工同步主控模式
- 半双工同步从动模式

为将引脚 RC6/TX/CK 和 RC7/RX/DT 配置为通用同步 / 异步收发器，需要：

- SPEN 位 （RCSTA<7>）必须置位 （=1），
- TRISC<6> 位必须清零 （=0），并且
- TRISC<7> 位必须置位 （=1）。

寄存器 16-1 显示发送状态和控制寄存器 （TXSTA），寄存器 16-2 显示接收状态和控制寄存器 （RCSTA）。

寄存器 16-1： TXSTA：发送状态和控制寄存器

R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D
bit 7							bit 0

位 7 **CSRC**：时钟源选择位
<u>异步模式：</u>
此位未用
<u>同步模式：</u>
1= 主控模式 （由 BRG 产生时钟）
0= 从动模式 （由外部时钟源提供时钟信号）

位 6 **TX9**：9 位发送使能位
1= 选择 9 位数据发送
0= 选择 8 位数据发送

位 5 **TXEN**：发送使能位
1= 允许发送
0= 禁止发送

注： 在 SYNC 模式下， SREN/CREN 位比 TXEN 位优先级高。

位 4 **SYNC**：USART 模式选择位
1= 同步模式
0= 异步模式

位 3 **未实现**：读作 0

位 2 **BRGH**：高速波特率选择位
<u>异步模式：</u>
1= 高速
0= 低速
<u>同步模式：</u>
在此模式下未使用此位

位 1 **TRMT**：发送移位寄存器状态位
1=TSR 空
0=TSR 满

位 0 **TX9D**：发送数据的第 9 位
可能是地址 / 数据位或奇偶校验位。

图注：			
R= 可读位	W= 可写位	U= 未实现位，读作 0	
- n= 上电复位时的值	1= 置位	0= 清零	x= 未知

寄存器 16-2： **RCSTA：接收状态和控制寄存器**

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

位 7 **SPEN**：串口使能位
1= 使能串口 （把 RX/DT 和 TX/CK 引脚配置为串口引脚）
0= 禁止串口

位 6 **RX9**：9 位接收使能位
1= 选择 9 位接收
0= 选择 8 位接收

位 5 **SREN**：单字节接收使能位
异步模式：
未用此位
同步主控模式
1= 允许接收单字节
0= 禁止接收单字节
在接收完成后该位被清零。
同步从动模式：
未用此位

位 4 **CREN**：连续接收使能位
异步模式：
1= 允许接收器
0= 禁止接收器
同步模式：
1= 允许连续接收，直到 CREN 使能位被清零 （CREN 位比 SREN 位优先级高）为止
0= 禁止连续接收

位 3 **ADDEN**：地址检测使能位
9 位异步模式 （RX9=1）：
1= 允许地址检测、使能中断及装入接收缓冲器
当 RSR<8> 置 1 时
0= 禁止地址检测，接收所有字节，第 9 位可作为奇偶校验位

位 2 **FERR**：帧出错标志位
1= 帧出错 （读 RCREG 寄存器可更新该位，并接收下一个有效字节）
0= 无帧错误

位 1 **OERR**：溢出错误位
1= 有溢出错误 （清零 CREN 位可将此位清零）
0= 无溢出错误

位 0 **RX9D**：接收数据的第 9 位
此位可作为地址 / 数据位或奇偶校验位，且必须由用户固件计算得到。

图注：			
R= 可读位	W= 可写位	U= 未实现位，读作 0	
-n= 上电复位时的值	1= 置位	0= 清零	x= 未知

16.1 USART 波特率发生器 （BRG）

波特率发生器 （Baud Rate Generator， BRG）支持 USART 的同步模式和异步模式。这是一个专用的 8 位波特率发生器。 SPBRG 寄存器控制着独立运行的 8 位定时器的周期。在异步模式下，BRGH 位（TXSTA<2>）也用于控制波特率。在同步模式下可忽略 BRGH 位。表 16-1 显示主控模式 （内部时钟）下，不同 USART 工作模式的波特率的计算公式。

给定目标波特率和 Fosc，可用表 16-1 中的公式计算出 SPBRG 寄存器的最接近的整数值。由此可计算出波特率的误差。

例 16-1 是给定下列条件时计算波特率误差的过程：

- Fosc=16 MHz
- 目标波特率 =9600
- BRGH=0
- SYNC=0

即使波特率时钟为低速，也最好能使用高速波特率公式（BRGH=1）。因为在某些情形下，公式 Fosc/(16(X + 1)) 可以降低波特率误差。

向 SPBRG 寄存器写入新值，会使 BRG 定时器复位（或清零）。这可确保 BRG 不需要等到定时器溢出就可以输出新的波特率。

16.1.1 采样

信号检测电路对 RC7/RX/DT 引脚采样三次，以判定 RX 引脚上出现的是高电平还是低电平。

例 16-1： 计算波特率误差

```
目标波特率        =  Fosc /  (64 (X + 1))
X 求解：
        X        =  ( (Fosc / 目标波特率 ) / 64 )-1
        X        =  ((16000000 / 9600) / 64)-1
        X        =   [25.042] = 25
计算得到的波特率  =   16000000 / (64 (25 + 1))
                 =   9615
误差              =   ( 计算得到的波特率 - 目标波特率 )
                               目标波特率
                 =   (9615-9600) / 9600
                 =   0.16%
```

表 16-1： 波特率计算公式

SYNC	BRGH = 0 （低速）	BRGH = 1 （高速）
0	（异步）波特率 = Fosc/(64(X+1))	波特率 = Fosc/(16(X+1))
1	（同步）波特率 = Fosc/(4(X+1))	N/A

图注： X 为 SPBRG 寄存器中的值 （0 到 255）

表 16-2： 与波特率发生器有关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
SPBRG	波特率发生器寄存器								0000 0000	0000 0000

图注： x = 未知， - = 未实现，读作 0。阴影单元表示 BRG 未使用。

表 **16-3**： 同步模式下的波特率

目标波特率 (Kbps)	Fosc = 40 MHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	33 MHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	25 MHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	20 MHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—	NA	—	—
9.6	NA	—	—	NA	—	—	NA	—	—	NA	—	—
19.2	NA	—	—	NA	—	—	NA	—	—	NA	—	—
76.8	76.92	+0.16	129	77.10	+0.39	106	77.16	+0.47	80	76.92	+0.16	64
96	96.15	+0.16	103	95.93	-0.07	85	96.15	+0.16	64	96.15	+0.16	51
300	303.03	+1.01	32	294.64	-1.79	27	297.62	-0.79	20	294.12	-1.96	16
500	500	0	19	485.30	-2.94	16	480.77	-3.85	12	500	0	9
高	10000	—	0	8250	—	0	6250	—	0	5000	—	0
低	39.06	—	255	32.23	—	255	24.41	—	255	19.53	—	255

目标波特率 (Kbps)	Fosc = 16 MHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	10 MHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	7.15909 MHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	5.0688 MHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—	NA	—	—
9.6	NA	—	—	NA	—	—	9.62	+0.23	185	9.60	0	131
19.2	19.23	+0.16	207	19.23	+0.16	129	19.24	+0.23	92	19.20	0	65
76.8	76.92	+0.16	51	75.76	-1.36	32	77.82	+1.32	22	74.54	-2.94	16
96	95.24	-0.79	41	96.15	+0.16	25	94.20	-1.88	18	97.48	+1.54	12
300	307.70	+2.56	12	312.50	+4.17	7	298.35	-0.57	5	316.80	+5.60	3
500	500	0	7	500	0	4	447.44	-10.51	3	422.40	-15.52	2
高	4000	—	0	2500	—	0	1789.80	—	0	1267.20	—	0
低	15.63	—	255	9.77	—	255	6.99	—	255	4.95	—	255

目标波特率 (Kbps)	Fosc = 4 MHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	3.579545 MHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	1 MHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	32.768 kHz 计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)
0.3	NA	—	—	NA	—	—	NA	—	—	0.30	+1.14	26
1.2	NA	—	—	NA	—	—	1.20	+0.16	207	1.17	-2.48	6
2.4	NA	—	—	NA	—	—	2.40	+0.16	103	2.73	+13.78	2
9.6	9.62	+0.16	103	9.62	+0.23	92	9.62	+0.16	25	8.20	-14.67	0
19.2	19.23	+0.16	51	19.04	-0.83	46	19.23	+0.16	12	NA	—	—
76.8	76.92	+0.16	12	74.57	-2.90	11	83.33	+8.51	2	NA	—	—
96	1000	+4.17	9	99.43	+3.57	8	83.33	-13.19	2	NA	—	—
300	333.33	+11.11	2	298.30	-0.57	2	250	-16.67	0	NA	—	—
500	500	0	1	447.44	-10.51	1	NA	—	—	NA	—	—
高	1000	—	0	894.89	—	0	250	—	0	8.20	—	0
低	3.91	—	255	3.50	—	255	0.98	—	255	0.03	—	255

表 16-4： 异步模式下的波特率 （BRGH = 0）

目标波特率 (Kbps)	Fosc = 40 MHz			33 MHz			25 MHz			20 MHz		
	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	2.40	-0.07	214	2.40	-0.15	162	2.40	+0.16	129
9.6	9.62	+0.16	64	9.55	-0.54	53	9.53	-0.76	40	9.47	-1.36	32
19.2	18.94	-1.36	32	19.10	-0.54	26	19.53	+1.73	19	19.53	+1.73	15
76.8	78.13	+1.73	7	73.66	-4.09	6	78.13	+1.73	4	78.13	+1.73	3
96	89.29	-6.99	6	103.13	+7.42	4	97.66	+1.73	3	104.17	+8.51	2
300	312.50	+4.17	1	257.81	-14.06	1	NA	—	—	312.50	+4.17	0
500	625	+25.00	0	NA	—	—	NA	—	—	NA	—	—
高	625	—	0	515.63	—	0	390.63	—	0	312.50	—	0
低	2.44	—	255	2.01	—	255	1.53	—	255	1.22	—	255

目标波特率 (Kbps)	Fosc = 16 MHz			10 MHz			7.15909 MHz			5.0688 MHz		
	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	1.20	+0.16	207	1.20	+0.16	129	1.20	+0.23	92	1.20	0	65
2.4	2.40	+0.16	103	2.40	+0.16	64	2.38	-0.83	46	2.40	0	32
9.6	9.62	+0.16	25	9.77	+1.73	15	9.32	-2.90	11	9.90	+3.13	7
19.2	19.23	+0.16	12	19.53	+1.73	7	18.64	-2.90	5	19.80	+3.13	3
76.8	83.33	+8.51	2	78.13	+1.73	1	111.86	+45.65	0	79.20	+3.13	0
96	83.33	-13.19	2	78.13	-18.62	1	NA	—	—	NA	—	—
300	250	-16.67	0	156.25	-47.92	0	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—	NA	—	—
高	250	—	0	156.25	—	0	111.86	—	0	79.20	—	0
低	0.98	—	255	0.61	—	255	0.44	—	255	0.31	—	255

目标波特率 (Kbps)	Fosc = 4 MHz			3.579545 MHz			1 MHz			32.768 kHz		
	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	波特率 (KHZ)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)
0.3	0.30	-0.16	207	0.30	+0.23	185	0.30	+0.16	51	0.26	-14.67	1
1.2	1.20	+1.67	51	1.19	-0.83	46	1.20	+0.16	12	NA	—	—
2.4	2.40	+1.67	25	2.43	+1.32	22	2.23	-6.99	6	NA	—	—
9.6	8.93	-6.99	6	9.32	-2.90	5	7.81	-18.62	1	NA	—	—
19.2	20.83	+8.51	2	18.64	-2.90	2	15.63	-18.62	0	NA	—	—
76.8	62.50	-18.62	0	55.93	-27.17	0	NA	—	—	NA	—	—
96	NA	—	—	NA	—	—	NA	—	—	NA	—	—
300	NA	—	—	NA	—	—	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—	NA	—	—
高	62.50	—	0	55.93	—	0	15.63	—	0	0.51	—	0
低	0.24	—	255	0.22	—	255	0.06	—	255	0.002	—	255

表 16-5： 异步模式下的波特率 （BRGH = 1）

目标波特率 (Kbps)	Fosc = 40 MHz			33 MHz			25 MHz			20 MHz		
	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	NA	—	—	NA	—	—
9.6	NA	—	—	9.60	-0.07	214	9.59	-0.15	162	9.62	+0.16	129
19.2	19.23	+0.16	129	19.28	+0.39	106	19.30	+0.47	80	19.23	+0.16	64
76.8	75.76	-1.36	32	76.39	-0.54	26	78.13	+1.73	19	78.13	+1.73	15
96	96.15	+0.16	25	98.21	+2.31	20	97.66	+1.73	15	96.15	+0.16	12
300	312.50	+4.17	7	294.64	-1.79	6	312.50	+4.17	4	312.50	+4.17	3
500	500	0	4	515.63	+3.13	3	520.83	+4.17	2	416.67	-16.67	2
高	2500	—	0	2062.50	—	0	1562.50	—	0	1250	—	0
低	9.77	—	255	8,06	—	255	6.10	—	255	4.88	—	255

目标波特率 (Kbps)	Fosc = 16 MHz			10 MHz			7.15909 MHz			5.0688 MHz		
	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	NA	—	—	NA	—	—	NA	—	—
2.4	NA	—	—	NA	—	—	2.41	+0.23	185	2.40	0	131
9.6	9.62	+0.16	103	9.62	+0.16	64	9.52	-0.83	46	9.60	0	32
19.2	19.23	+0.16	51	18.94	-1.36	32	19.45	+1.32	22	18.64	-2.94	16
76.8	76.92	+0.16	12	78.13	+1.73	7	74.57	-2.90	5	79.20	+3.13	3
96	100	+4.17	9	89.29	-6.99	6	89.49	-6.78	4	105.60	+10.00	2
300	333.33	+11.11	2	312.50	+4.17	1	447.44	+49.15	0	316.80	+5.60	0
500	500	0	1	625	+25.00	0	447.44	-10.51	0	NA	—	—
高	1000	—	0	625	—	0	447.44	—	0	316.80	—	0
低	3.91	—	255	2.44	—	255	1.75	—	255	1.24	—	255

目标波特率 (Kbps)	Fosc = 4 MHz			3.579545 MHz			1 MHz			32.768 kHz		
	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)	计算波特率 (Kbps)	误差 (%)	SPBRG 值 (十进制)
0.3	NA	—	—	NA	—	—	0.30	+0.16	207	0.29	-2.48	6
1.2	1.20	+0.16	207	1.20	+0.23	185	1.20	+0.16	51	1.02	-14.67	1
2.4	2.40	+0.16	103	2.41	+0.23	92	2.40	+0.16	25	2.05	-14.67	0
9.6	9.62	+0.16	25	9.73	+1.32	22	8.93	-6.99	6	NA	—	—
19.2	19.23	+0.16	12	18.64	-2.90	11	20.83	+8.51	2	NA	—	—
76.8	NA	—	—	74.57	-2.90	2	62.50	-18.62	0	NA	—	—
96	NA	—	—	111.86	+16.52	1	NA	—	—	NA	—	—
300	NA	—	—	223.72	-25.43	0	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—	NA	—	—
高	250	—	0	55.93	—	0	62.50	—	0	2.05	—	0
低	0.98	—	255	0.22	—	255	0.24	—	255	0.008	—	255

16.2 USART 异步工作模式

在异步工作模式下，USART 采用标准的不归零（NRZ）格式（一位起始位、8 位或 9 位数据位和一位停止位）。最常用的数据格式是 8 位。片内专用的 8 位波特率发生器可用于根据振荡器产生标准的波特率频率。 USART 首先发送和接收最低有效位（LSb）。USART 的发送器和接收器在功能上是独立的，但采用相同的数据格式和波特率。根据 BRGH 位 （TXSTA<2>）的值，对于波特率发生器产生的时钟，其移位速率有两种：x16 或 x64。USART 硬件不支持奇偶校验，但可以用软件实现（奇偶校验位是第 9 个数据位）。在休眠状态下，异步模式将被停止。

通过对 SYNC 位 （TXSTA<4>）清零，可选择 USART 异步工作模式。

USART 异步模块有以下重要的组成部分：

- 波特率发生器
- 采样电路
- 异步发送器
- 异步接收器

16.2.1 USART 异步发送器

图 16-1 是 USART 发送器结构框图。发送器的核心是发送（串行）移位寄存器（TSR）。发送移位寄存器从读 / 写发送缓冲器 TXREG 获得要发送的数据。TXREG 寄存器中的数据由软件写入。前一次装入的停止位发送出去后， TSR 寄存器才装入数据。停止位一发送出去，TXREG 寄存器中的新数据（如果有的话）就会被装入 TSR。一旦把 TXREG 寄存器中的数据送入 TSR 寄存器（在一个 TCY 周期内），TXREG 寄存器就变为空，同时发送中断标志位 TXIF（PIR1<4>）会置 1。可以通过将 TXIE 使能位 （PIE1<4>）置 1 或清零来使能或禁止该中断。不管 TXIE 使能位的状态如何，都将把中断标志位 TXIF 置 1，且不能用软件清零。只有把新数据写入 TXREG 寄存器后，才会复位 TXIF 位。 TXIF 标志位表示 TXREG 寄存器的状态，而 TRMT 位 （TXSTA<1>）表示 TSR 寄存器的状态。TRMT 状态位是一个只读位，当 TSR 寄存器为空时被置 1。TRMT 位与任何中断逻辑都没有联系，所以要确定 TSR 寄存器是否为空，用户必须对该位进行查询。

> **注** 1：TSR 寄存器并未映射到数据存储器中，因此用户不能直接访问它。
>
> 2：当使能位 TXEN 置 1 时，也会将标志位 TXIF 置 1。

要设置异步发送模式：

1. 选择合适的波特率对 SPBRG 寄存器进行初始化。如果需要高速波特率，将BRGH置1（第 16.1 节）。
2. 将 SYNC 位清零，SPEN 位置 1，使能异步串行口。
3. 若需要中断，将使能位 TXIE 置 1。
4. 若需要发送 9 位数据，将发送位 TX9 置 1。可以用作地址 / 数据位。
5. 将 TXEN 位置 1 来使能发送方式，同时将 TXIF 位置 1。
6. 若选择发送 9 位数据，要先把第 9 位装入 TX9D 位。
7. 把数据装入 TXREG 寄存器 （启动发送）。

> **注：** 将数据送入发送缓冲器TXREG时不会立刻清零 TXIF。在装入指令后的第二个指令周期内标志位变为有效。



图 16-1： USAR 发送结构框图

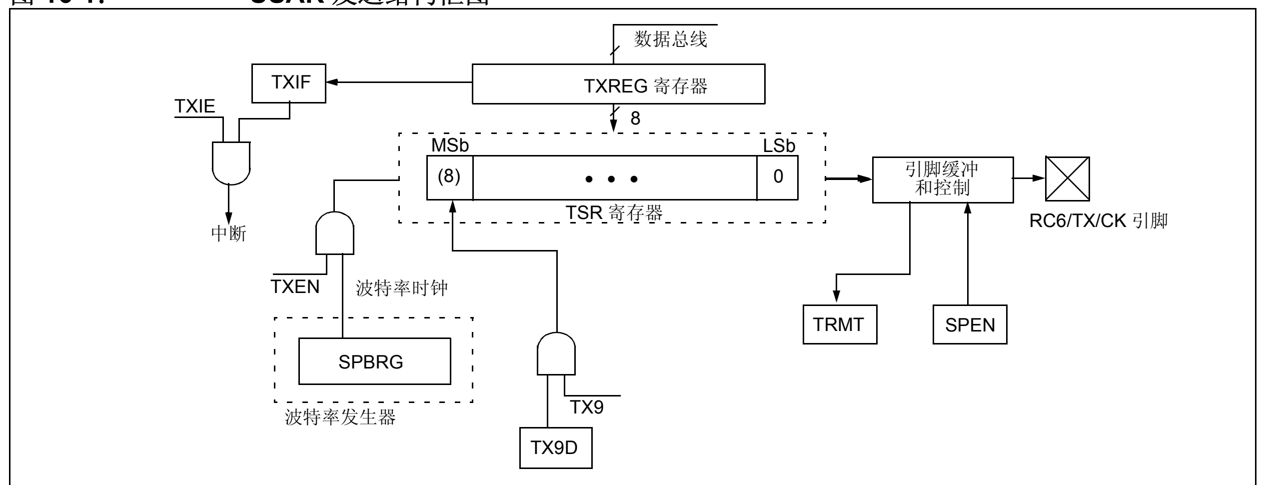


图 16-2： 异步发送

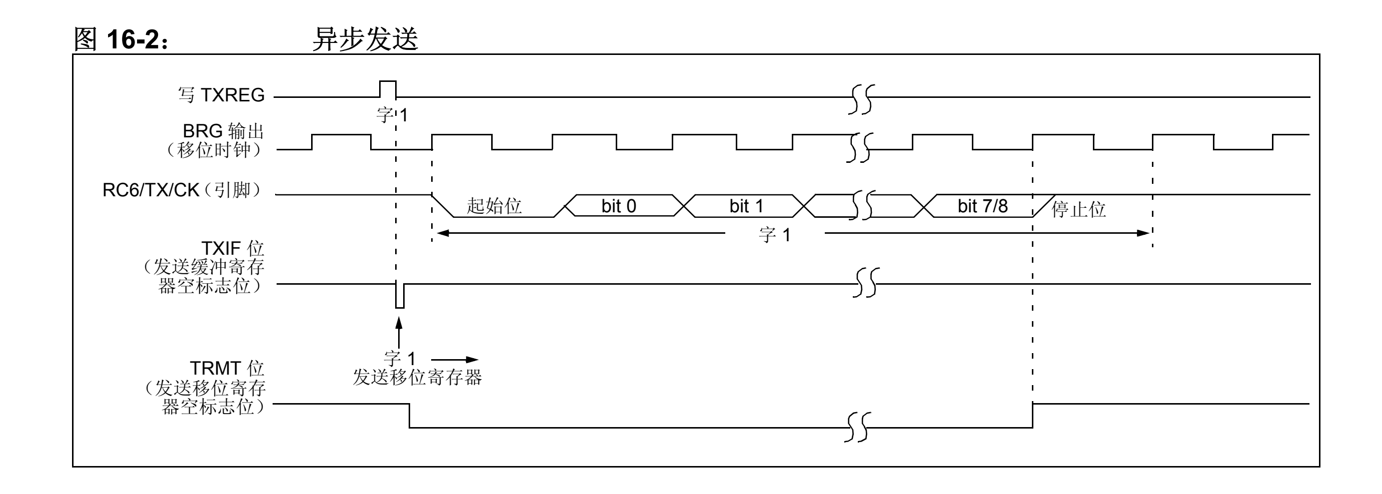
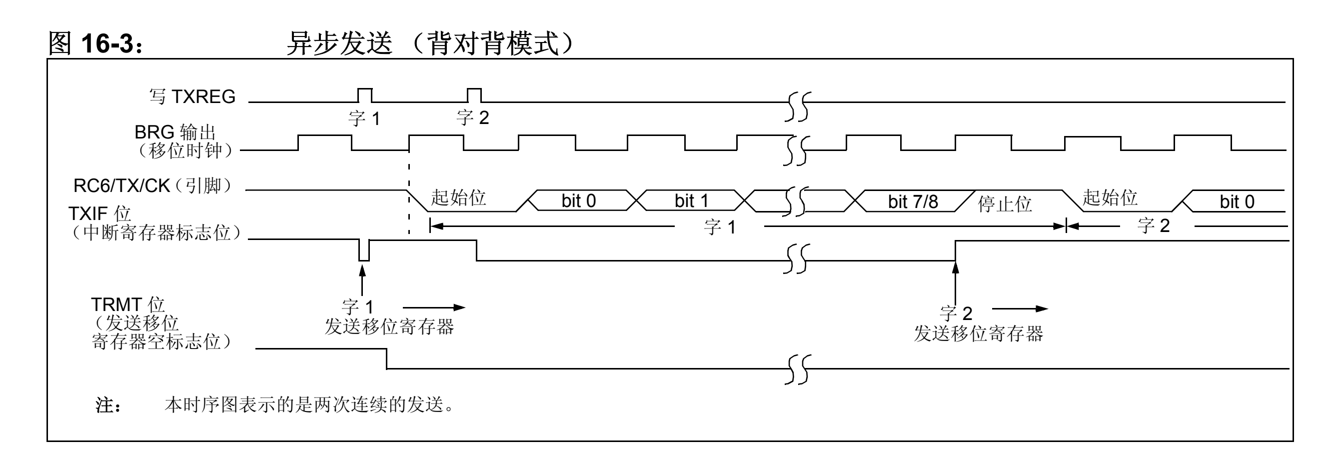


图 16-3： 异步发送 （背对背模式）



注： 本时序图表示的是两次连续的发送。

表 16-6： 与异步发送有关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART 发送寄存器								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	波特率发生器寄存器								0000 0000	0000 0000

图注： x = 未知， - = 未实现，读作 0。
阴影单元表示异步发送未使用。

注 1：在 PIC18F2X 器件上保留了 PSPIF、 PSPIE 和 PSPIP 位；这些位始终保持为零。

16.2.2 USART 异步接收器

图 16-4 是接收器结构框图。在 RC7/RX/DT 引脚上接收数据，并驱动数据恢复电路。数据恢复电路其实是一个以波特率 x16 的速率工作的高速移位寄存器，而主接收串行移位寄存器以比特率或 FOSC 工作。此模式通常用于 RS-232 系统中。

要设置异步接收模式：

1. 选择合适的波特率对 SPBRG 寄存器进行初始化。如果需要高速波特率，将BRGH置1（第 16.1 节）。
2. 将 SYNC 位清零，SPEN 位置 1，使能异步串行口。
3. 若需要中断，将使能位 RCIE 置 1。
4. 若需要接收 9 位数据，将 RX9 位置 1。
5. 将 CREN 位置 1 来使能接收方式。
6. 当接收完成后，将把中断标志位 RCIF 置 1。如果此时使能位 RCIE 已置 1，便产生中断。
7. 读取 RCSTA 寄存器获取第 9 位数据（如果已使能），并判断在接收操作中是否发生错误。
8. 读取 RCREG 寄存器来读取接收到的 8 位数据。
9. 如果发生错误，通过将使能位 CREN 清零来清除错误。
10. 如果使用中断，确保将 INTCON 寄存器的 GIE 位和 PEIE 位（INTCON<7:6>）置 1。

16.2.3 将 9 位方式用于地址检测

此模式通常用在 RS-485 系统中。要设置使能了地址检测的异步接收模式：

1. 选择合适的波特率对 SPBRG 寄存器进行初始化。如果需要高速波特率，将 BRGH 置 1。
2. 将 SYNC 位清零，SPEN 位置 1，使能异步串行口。
3. 若需要中断，将使能位 RCEN 置 1，用 RCIP 位选择需要的优先级。
4. 将 RX9 位置 1 来使能 9 位接收方式。
5. 将 ADDEN 位置 1 来使能地址检测。
6. 将 CREN 位置 1 来使能接收方式。
7. 当接收完成后，将把 RCIF 位置 1。如果 RCIE 位和 GIE 位都已置 1，则应答中断。
8. 读取 RCSTA 寄存器以及数据的第 9 位（如果适用），以判断在接收过程中是否发生错误。
9. 读取 RCREG 寄存器以判断是否正在对器件进行寻址。
10. 如果发生错误，将 CREN 位清零。
11. 如果已经对器件进行了寻址，将ADDEN位清零，允许将所有收到的数据写入接收缓冲器并中断 CPU。

图 16-4：　USART 接收结构框图

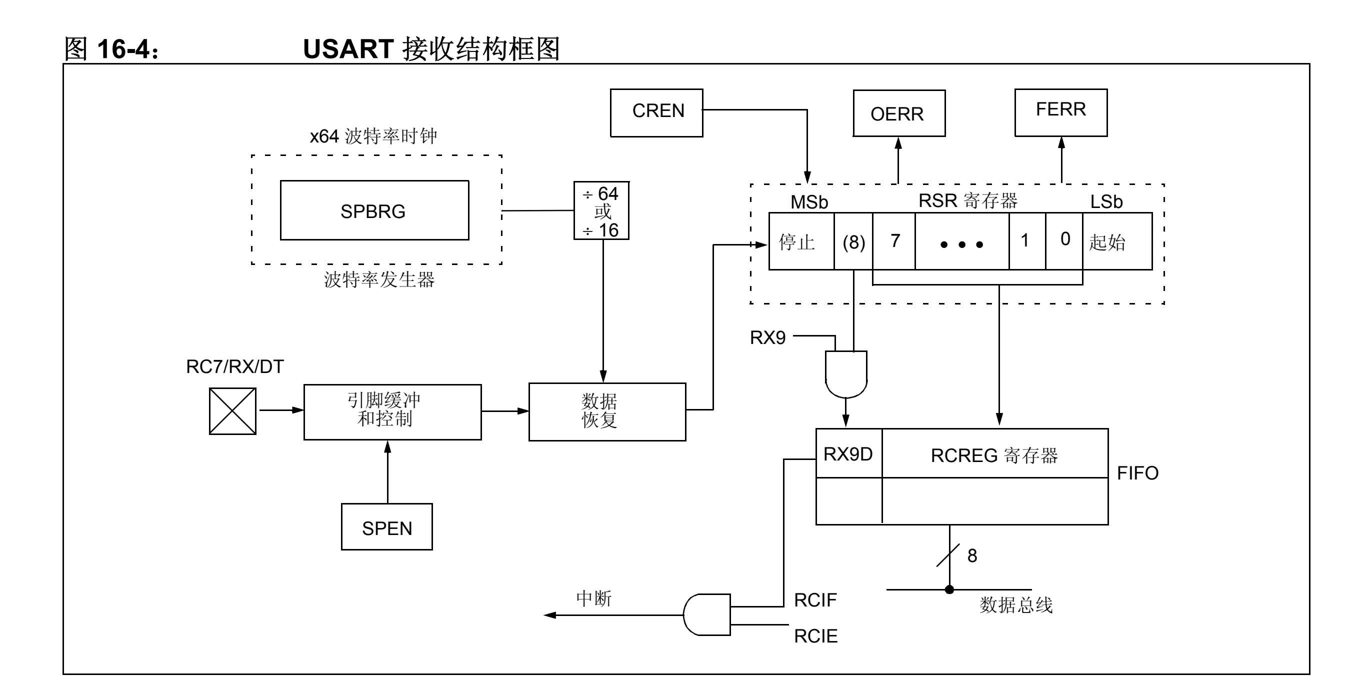


图 16-5： 异步接收

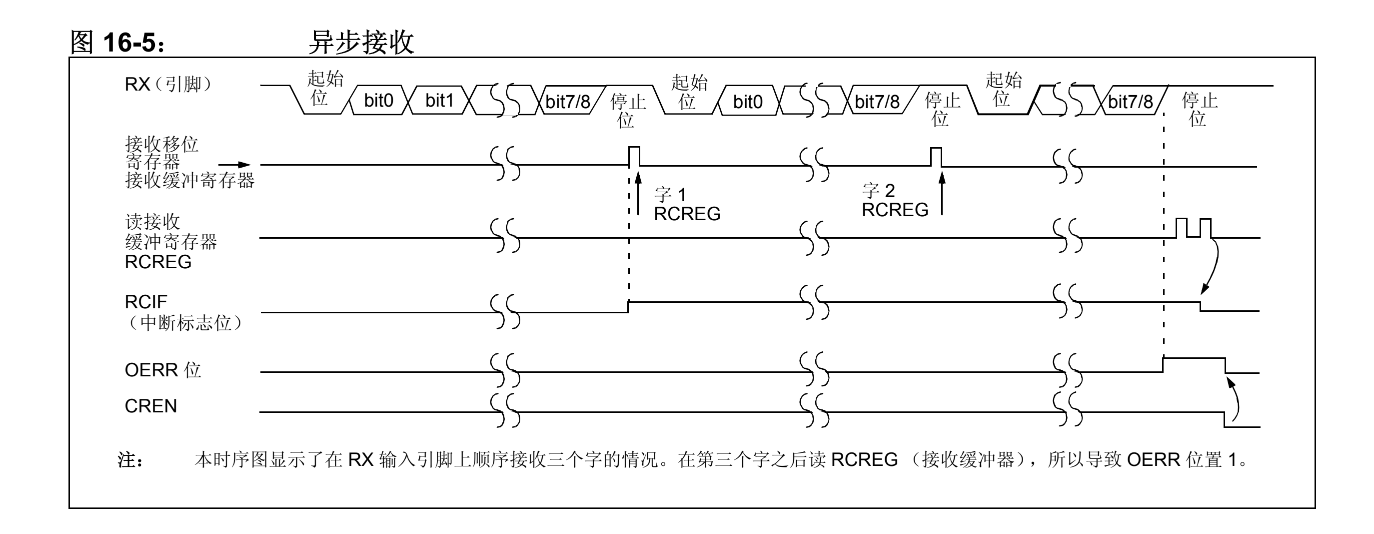


表 16-7： 与异步接收有关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
INTCON	GIE/GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART 接收寄存器								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	波特率发生器寄存器								0000 0000	0000 0000

图注： x = 未知， - = 未实现，读作 0。
阴影单元表示异步接收未使用。

注 1：在 PIC18F2X 器件上保留了 PSPIF、 PSPIE 和 PSPIP 位；这些位始终保持为零。

16.3 USART 同步主控模式

在同步主控模式下，数据以半双工方式传输（即收发不同时进行）。在发送数据时，禁止接收数据，反之亦然。把 SYNC 位（TXSTA<4>）置 1 就可以进入同步工作模式。此外，应把 SPEN 使能位（RCSTA<7>）置 1，以将 RC6/TX/CK 和 RC7/RX/DT I/O 引脚分别配置为时钟线 CK 和数据线 DT。主控模式意味着处理器在 CK 线上发送主控时钟信号。把 CSRC 位（TXSTA<7>）置 1 可进入主控模式。

16.3.1 USART 同步主控发送

图 16-1 是 USART 发送器结构框图。发送器的核心是发送（串行）移位寄存器（TSR）。发送移位寄存器从读/写发送缓冲寄存器TXREG获取要发送的数据。TXREG寄存器中的数据由软件写入。要等待TSR发送完上次装入数据的最后一位，才会将 TXREG 中的数据装入 TSR中。TSR 中的最后一位一发送完，TSR 就从 TXREG 中装入新的数据（如果有数据的话）。当 TXREG 寄存器把数据送入 TSR（在一个 TCYCLE 内）后，TXREG 变为空，同时会把中断标志位 TXIF（PIR1<4>）置 1。对使能位 TXIE（PIE1<4>）置 1/ 清零，可允许 / 禁止中断。不管使能位 TXIE 的状态如何，都要将标志位 TXIF置 1，并且 TXIF 位不能用软件清零。只有把新数据写入TXREG 寄存器后，才能复位 TXIF 位。TXIF 标志位表明 TXREG 寄存器的状态，而 TRMT 位（TXSTA<1>）表明 TSR 寄存器的状态。TRMT 位是一个只读位，当TSR 为空时，TRMT 置 1。没有任何中断逻辑与 TRMT位有联系，所以只能通过对 TRMT 位查询来判断 TSR寄存器是否为空。TSR 并未映射到数据存储器中，所以用户不能直接访问它。

要设置同步主控发送模式：

1. 选择合适的波特率对 SPBRG 寄存器进行初始化（第 16.1 节）。
2. 将 SYNC、SPEN 和 CSRC 位置 1，使能同步主控串行口。
3. 若需要中断，将使能位 TXIE 置 1。
4. 若需要传送 9 位数据，将 TX9 位置 1。
5. 将 TXEN 位置 1 来使能发送方式。
6. 若选择发送 9 位数据，应先把第 9 位装入 TX9D位。
7. 把数据送入 TXREG 寄存器来启动发送。

注：	将数据送入发送缓冲器TXREG时不会立刻清零 TXIF。在装入指令后的第二个指令周期内标志位变为有效位。

表 16-8： 与同步主控发送有关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF[1]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE[1]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP[1]	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART 发送寄存器								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	波特率发生器寄存器								0000 0000	0000 0000

图注： x = 未知，- = 未实现，读作 0。
阴影单元表示同步主控发送未使用。
注 1：在 PIC18F2X 器件上保留了 PSPIF、PSPIE 和 PSPIP 位；这些位始终保持为零。

图 16-6： 同步发送

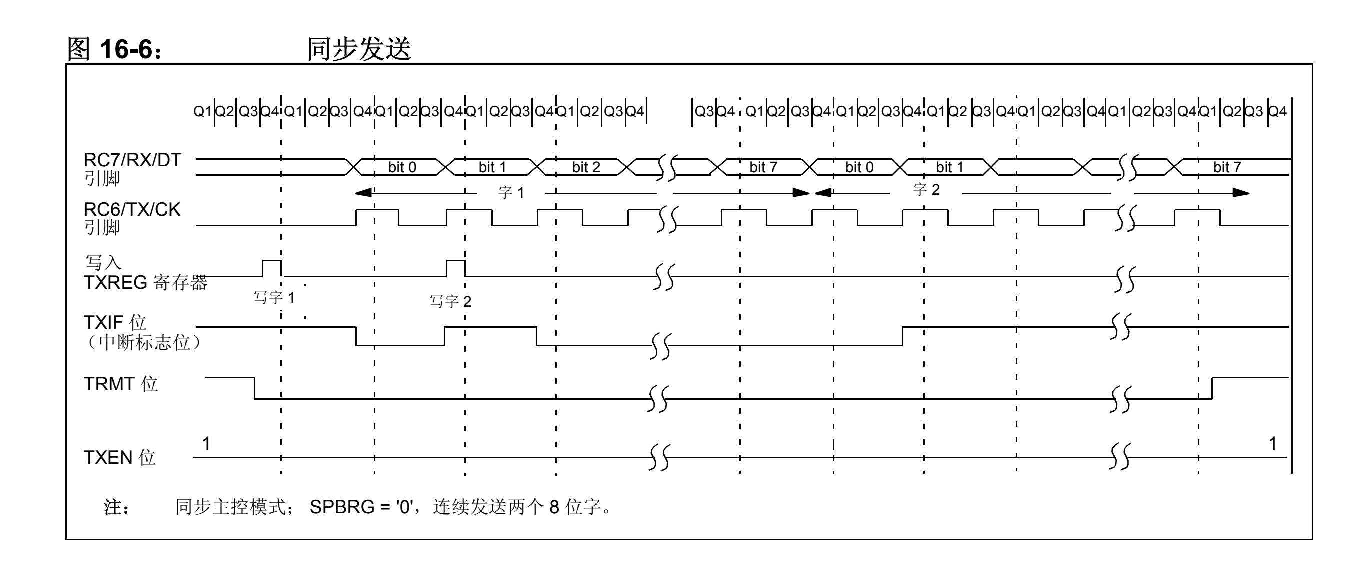
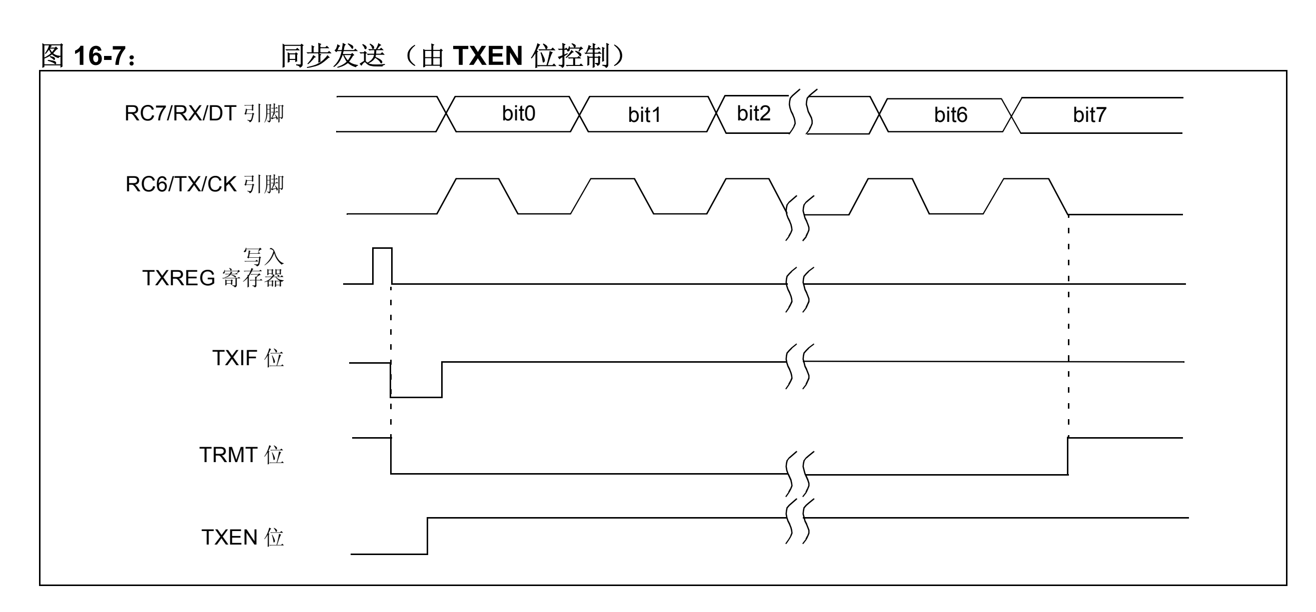


图 16-7： 同步发送 （由 **TXEN** 位控制）



16.3.2 USART 同步主控接收

一旦选择了同步模式后，只要把使能位 SREN（RCSTA<5>）或 CREN 位（RCSTA<4>）置 1，即可使能接收。在时钟的下降沿采样 RC7/RX/DT 引脚上的数据。如果 SREN 置了 1，则仅接收一个字。如果 CREN 置了 1，则可连续地接收数据，直到 CREN 位被清零。如果两个位都被置1，则CREN位优先而进行连续接收。

要设置同步主控接收模式：

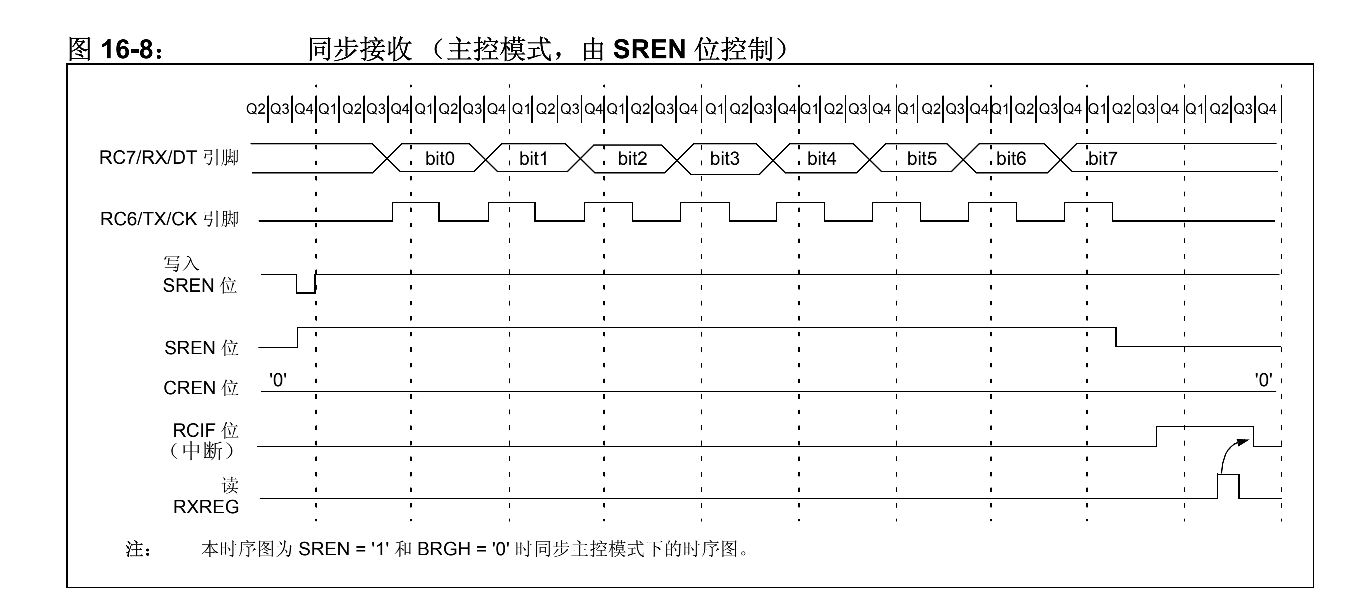
1. 选择合适的波特率对 SPBRG 寄存器进行初始化（第 16.1 节）。
2. 将 SYNC、SPEN 和 CSRC 位置 1，使能同步主控串行口。
3. 确保 CREN 和 SREN 位清零。
4. 若需要中断，将使能位 RCIE 置 1。
5. 如果需要接收 9 位数据，将 RX9 位置 1。
6. 若需要单字节接收，将 SREN 位置 1。若需要连续接收，将 CREN 位置 1。
7. 当接收完成后，将中断标志位 RCIF 置 1，如果此时使能位 RCIE 已置 1，便产生中断。
8. 读取 RCSTA 寄存器获取第 9 位数据（如果已使能），并判断在接收操作中是否发生错误。
9. 读取 RCREG 寄存器来读取收到的 8 位数据。
10. 如果发生错误，通过将 CREN 位清零来清除错误。
11. 如果使用中断，确保将 INTCON 寄存器的 GIE 位和 PEIE 位（INTCON<7:6>）置 1。

表 16-9： 与同步主控接收方式有关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART 接收寄存器								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	波特率发生器寄存器								0000 0000	0000 0000

图注： x = 未知， - = 未实现，读作 0。阴影单元表示同步主控接收未使用。
注 1：在 PIC18F2X 器件上保留了 PSPIF、 PSPIE 和 PSPIP 位；这些位始终保持为零。

图 16-8： 同步接收（主控模式，由 **SREN** 位控制）



16.4 USART 同步从动模式

同步从动模式与主控模式不同，其移位时钟信号是在RC6/TX/CK 引脚上由外部提供（而主控模式是由内部提供移位时钟）。这样就允许器件在休眠状态下发送或接收数据。把 CSRC 位（TXSTA<7>）清零即可进入从动模式。

16.4.1 USART 同步从动发送

除了休眠状态外，同步主控模式和从动模式的工作方式是一样的。

如果向 TXREG 写入 2 个字，然后执行 SLEEP 指令，则：

a) 第一个字立即传送到 TSR 寄存器进行发送。
b) 第二个字仍保存在 TXREG 寄存器中。
c) 不会将 TXIF 标志位置 1。
d) 当第一个字已移出 TSR 后，TXREG 寄存器将第二个字送入 TSR，同时将 TXIF 标志位置 1。
e) 如果使能位 TXIE 已置 1，中断将把芯片从休眠状态唤醒。如果使能全局中断，那么程序就跳转到中断向量。

要设置同步从动发送模式：

1. 将 SYNC 和 SPEN 位置 1，清零 CSRC 位，来使能同步从动串口。
2. 将 CREN 和 SREN 位清零。
3. 若需要中断，将使能位 TXIE 置 1。
4. 若要发送 9 位数据，将 TX9 位置 1。
5. 将使能位 TXEN 置 1 来使能发送方式。
6. 若选择发送 9 位数据，应先把第 9 位装入 TX9D 位。
7. 把数据送入 TXREG 寄存器启动发送。
8. 如果使用中断，确保将 INTCON 寄存器的 GIE 位和 PEIE 位（INTCON<7:6>）置 1。

表 16-10： 与同步从动发送有关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF[(1)]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE[(1)]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP[(1)]	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	USART 发送寄存器								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	波特率发生器寄存器								0000 0000	0000 0000

图注：x = 未知，- = 未实现，读作 0。
阴影单元表示同步从动发送未使用。

注 1：在 PIC18F2X 器件上保留了 PSPIF、 PSPIE 和 PSPIP 位；这些位始终保持为零。

16.4.2 USART 同步从动接收

除了休眠状态外，同步主控模式和从动模式的工作方式是一样的，另外，在从动模式下“不考虑”SREN 位。

如果在执行 SLEEP 指令之前已使能接收模式（即把 CREN 位置 1），那么在休眠状态下仍可以接收数据字。当接收完一个数据字后， RSR 寄存器将把数据送入 RCREG 寄存器，并且如果 RCIE 使能位已置 1，则产生的中断将芯片从休眠状态中唤醒。如果使能了全局中断，那么程序就跳转到中断向量。

要设置同步从动接收模式：

1. 将 SYNC 和 SPEN 位置 1，清零 CSRC 位，来使能同步从动串口。
2. 若需要中断，将使能位 RCIE 置 1。
3. 若需要接收 9 位数据，将 RX9 位置 1。
4. 将使能位 CREN 置 1 来使能接收方式。
5. 接收完成后，将把标志位 RCIF 置 1。如果此时中断位 RCIE 已置 1，将产生中断。
6. 读取 RCSTA 寄存器获取第 9 位数据（如果已使能），并判断在接收操作中是否发生错误。
7. 读取 RCREG 寄存器来读取接收到的 8 位数据。
8. 如果发生错误，通过将 CREN 位清零来清除错误。
9. 如果使用中断，确保将 INTCON 寄存器的 GIE 位和 PEIE 位 （INTCON<7:6>）置 1。

表 16-11： 与同步从动接收有关的寄存器

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF[(1)]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE[(1)]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP[(1)]	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	USART 接收寄存器								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	波特率发生器寄存器								0000 0000	0000 0000

图注： x = 未知， - = 未实现，读作 0。
阴影单元表示同步从动接收未使用。

注 1：在 PIC18F2X 器件上保留了 PSPIF、 PSPIE 和 PSPIP 位；这些位始终保持为零。

17.0 兼容型 10 位模数转换器 （A/D） 模块

PIC18F2X2 器件的模数 （Analog-to-Digital， A/D）转换器模块有 5 个输入通道，而 PIC18F4X2 器件的 A/D 模块有8个输入通道。该模块包含ADCON0和ADCON1 寄存器定义，它们与中档系列 A/D 模块兼容。

A/D 模块能将一个模拟输入信号转换成相应的 10 位数字信号。

A/D 模块有 4 个寄存器。这些寄存器是：

- A/D 结果高位寄存器 （ADRESH）
- A/D 结果低位寄存器 （ADRESL）
- A/D 控制寄存器 0 （ADCON0）
- A/D 控制寄存器 1 （ADCON1）

ADCON0 寄存器（如寄存器 17-1 所示）控制 A/D 模块的操作。ADCON1 寄存器（如寄存器 17-2 所示）可以配置端口引脚的功能。

寄存器 17-1： ADCON0 寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	R/W-0
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON
bit 7							bit 0

bit 7-6 **ADCS1:ADCS0**：A/D 转换时钟选择位 （用**粗体**表示 ADCON0 位）

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	时钟转换
0	**00**	FOSC/2
0	**01**	FOSC/8
0	**10**	FOSC/32
0	**11**	FRC （来自内部 A/D RC 振荡器的时钟）
1	**00**	FOSC/4
1	**01**	FOSC/16
1	**10**	FOSC/64
1	**11**	FRC （来自内部 A/D RC 振荡器的时钟）

bit 5-3 **CHS2:CHS0**：模拟通道选择位
000 = 通道 0，（AN0）
001 = 通道 1，（AN1）
010 = 通道 2，（AN2）
011 = 通道 3，（AN3）
100 = 通道 4，（AN4）
101 = 通道 5，（AN5）
110 = 通道 6，（AN6）
111 = 通道 7，（AN7）

注： PIC18F2X2 器件没有实现所有的 8 个 A/D 通道，未实现的选项被保留。不要选择未实现的通道。

bit 2 **GO/$\overline{\text{DONE}}$**：A/D 转换状态位
当 ADON=1 时：
1 = 正在进行 A/D 转换 （将该位置 1 则启动 A/D 转换， A/D 转换结束后该位由硬件自动清零）
0 = 未进行 A/D 转换

bit 1 **未实现**：读作 0

bit 0 **ADON**：A/D 模块使能位
1=A/D 转换器模块上电
0=A/D 转换器关闭，不消耗工作电流

图注：			
R= 可读位	W= 可写位	U= 未实现位，读作 0	
-n= 上电复位时的值	1= 置位	0= 清零	x= 未知位

寄存器 17-2： **ADCON1 寄存器**

R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM**：A/D 结果格式选择位
1 = 右对齐，ADRESH 寄存器的高 6 位读作 0
0 = 左对齐，ADRESL 寄存器的低 6 位读作 0

bit 6 **ADCS2**：A/D 转换时钟选择位 （用**粗体**表示 ADCON1 位）

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	时钟转换
0	00	FOSC/2
0	01	FOSC/8
0	10	FOSC/32
0	11	FRC （来自内部 A/D RC 振荡器的时钟）
1	00	FOSC/4
1	01	FOSC/16
1	10	FOSC/64
1	11	FRC （来自内部 A/D RC 振荡器的时钟）

bit 5-4 **未实现**：读作 0

bit 3-0 **PCFG3:PCFG0**：A/D 端口配置控制位

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

A= 模拟输入 D= 数字 I/O
C/R= 模拟输入通道编号 /A/D 参考电压源编号

图注：			
R= 可读位	W= 可写位	U= 未实现位，读作 0	
-n= 上电复位时的值	1= 置位	0= 清零	x= 未知位

注： 器件复位时，与模拟功能 （ANx）复用的端口引脚强制变为模拟输入。

模拟参考电压可通过软件选择为器件的正电源电压和负电源电压（V_{DD} 和 V_{SS}）或 RA3/AN3/VREF+ 引脚和 RA2/AN2/VREF- 引脚上的电压电平。

A/D 转换器具备在器件处于休眠状态下仍能工作的独特功能。要使 A/D 模块在休眠状态下运行，A/D 转换时钟必须来自于 A/D 模块内部的 RC 振荡器。

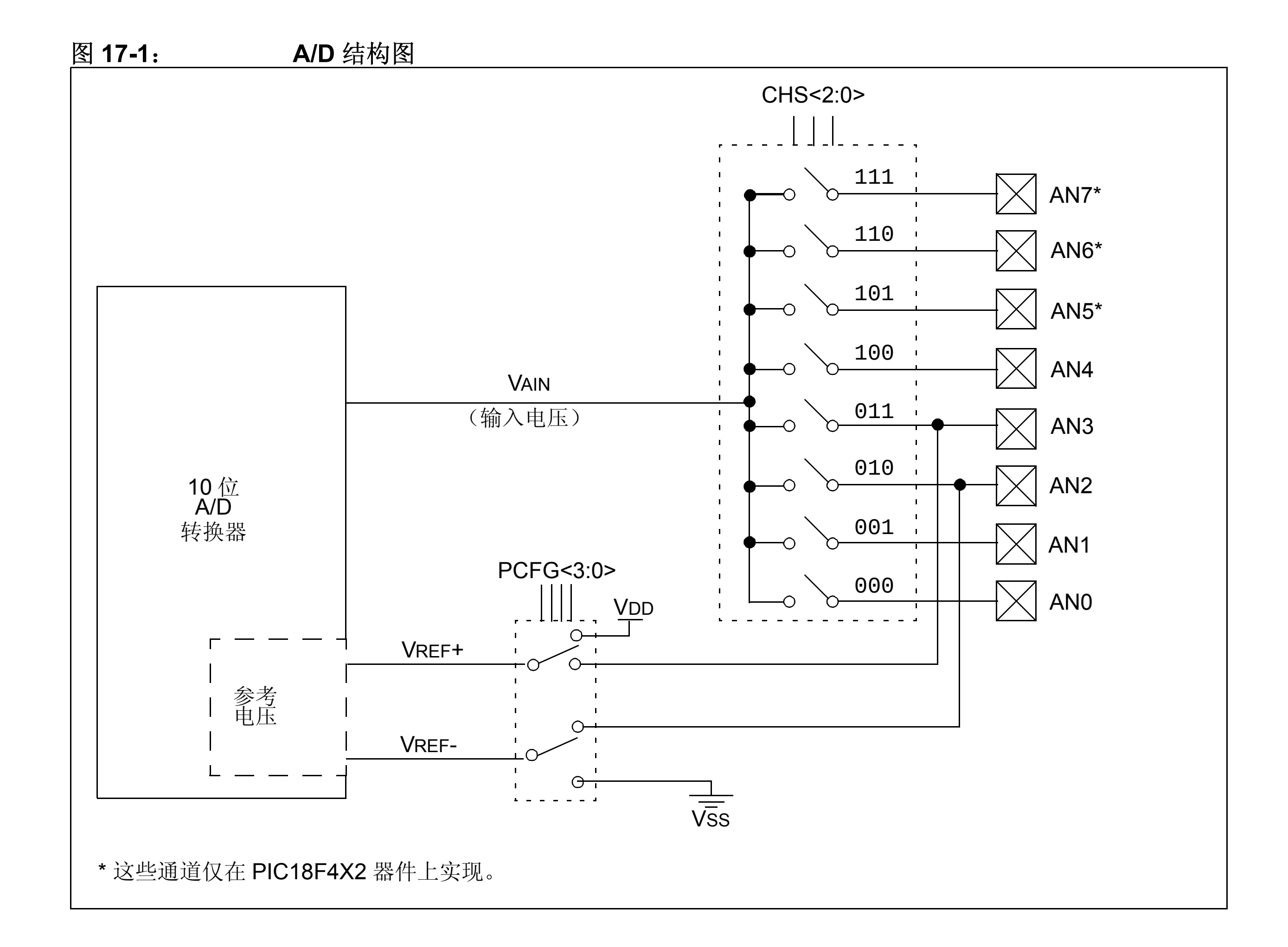
采样保持输出作为转换器的输入，A/D 转换器采用逐次逼近法得到转换结果。

器件复位迫使所有寄存器进入复位状态，同时强制关断 A/D 模块并中止任何转换。

每个和 A/D 转换器相关的端口引脚都可以配置成模拟输入（RA3 也可以作为参考电压）或数字 I/O。

ADRESH 和 ADRESL 寄存器中保存了 A/D 转换的结果。当 A/D 转换完成之后，转换结果装入 ADRESH/ADRESL 寄存器，清零 GO/DONE 位（ADCON0<2>），并将 A/D 中断标志位 ADIF 置 1。A/D 模块的结构图如图 17-1 所示。

图 17-1： A/D 结构图



上电复位时， ADRESH/ADRESL 寄存器中的值保持不变。上电复位后， ADRESH/ADRESL 寄存器中的值不确定。

按预期地配置好 A/D 模块后，在启动转换前必须先选择 A/D 转换通道。模拟输入通道相应的 TRIS 位必须设置为输入。采集时间的确定请参见第 17.1节。经过了这一采集时间之后，即可开始 A/D 转换。按照以下步骤进行 A/D 转换：

1. 配置 A/D 模块：
   - 配置模拟引脚、参考电压和数字 I/O （ADCON1）
   - 选择 A/D 输入通道 （ADCON0）
   - 选择 A/D 转换时钟 （ADCON0）
   - 打开 A/D 模块 （ADCON0）
2. 需要时，配置 A/D 中断：
   - 将 ADIF 位清零
   - 将 ADIE 位置 1
   - 将 GIE 位置 1
   - 将 PEIE 位置 1
3. 等待所需的采集时间。
4. 启动 A/D 转换：
   - 将 GO/$\overline{\text{DONE}}$ 位置 1 （ADCON0）
5. 等待 A/D 转换完成，通过以下两种方法之一可判断转换是否完成：
   - 轮询 GO/$\overline{\text{DONE}}$ 位是否被清零 （中断禁止）

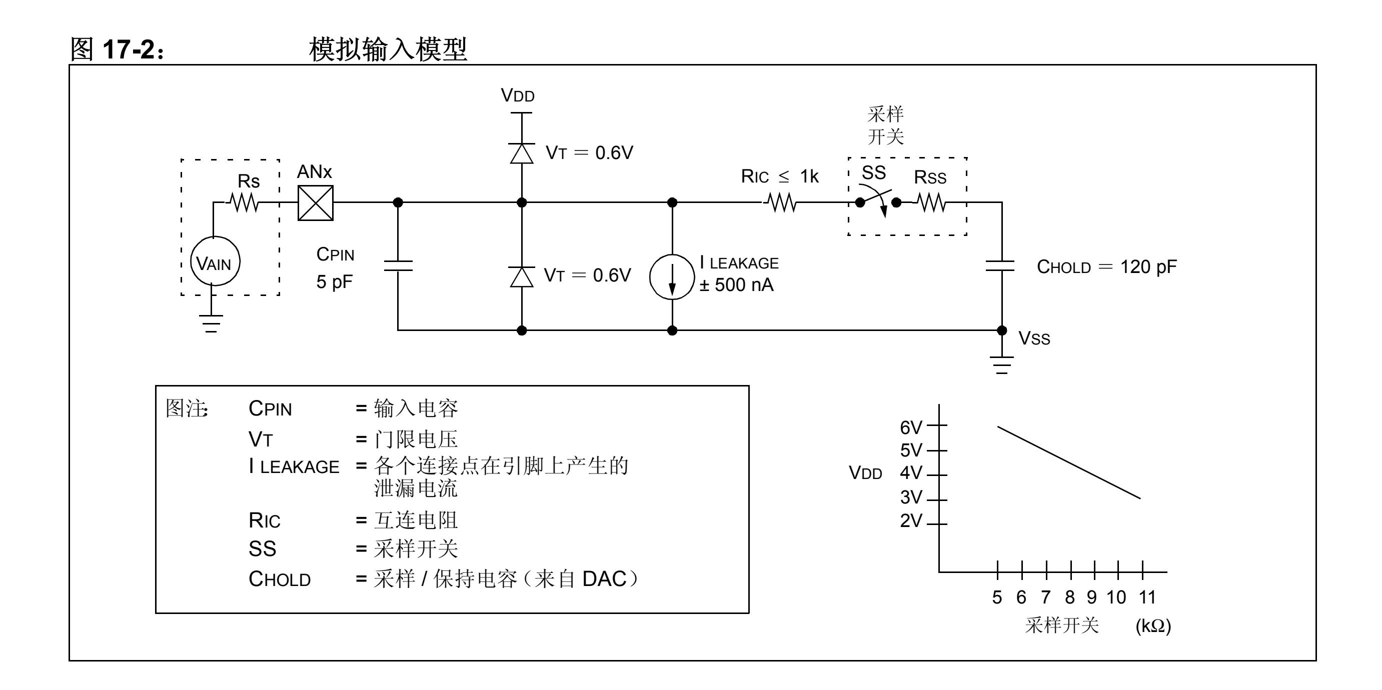
   或者
   - 等待 A/D 转换中断
6. 读取 A/D 结果寄存器（ADRESH/ADRESL），需要时将 ADIF 位清零。
7. 要再次进行 A/D 转换，根据要求转入步骤 1 或步骤 2。将每一位的 A/D 转换时间定义为 T_{AD}。在下一次采集开始前至少需要等待 $2T_{AD}$。

17.1 A/D 采集要求

为了使 A/D 转换达到规定精度，必须让充电保持电容（C_{HOLD}）充满至输入通道的电压电平。图 17-2 显示了模拟输入模型。模拟信号的源阻抗 （R_S）和内部采样开关阻抗 （R_{SS}）直接影响电容 C_{HOLD} 所需的充电时间。采样开关阻抗 （R_{SS}）随器件电压 （V_{DD}）变化。源阻抗影响模拟输入通道的偏置电压（因引脚上存在泄漏电流）。**建议模拟信号源的最大阻抗为 2.5 k** Ω。选择（改变）模拟输入通道后，必须完成模拟信号的采集才能开始 A/D 转换。

注：	当开始转换时，保持电容从输入引脚断开。

图 17-2： 模拟输入模型



要计算最小采集时间，可使用公式 17-1。该公式假设误差为 1/2 LSb（即 A/D 的 1024 步）。 1/2 LSb 误差是 A / D 模块达到规定分辨率的最大允许误差。

公式 17-1： 采集时间

$$T_{ACQ} = \text{放大器的建立时间＋保持电容器充电时间＋温度系数}$$
$$= T_{AMP} + T_C + T_{COFF}$$

公式 17-2： **A/D** 转换最小充电时间

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$
或
$$T_C = -(120\ pF)(1\ k\Omega + R_{SS} + R_S)\ \ln(1/2048)$$

例 17-1 显示了如何计算所需的最小采集时间 T_{ACQ}。该计算基于以下应用系统假定：

- C_{HOLD} =120 pF
- Rs =2.5 kΩ
- 转换误差 ≤1/2 LSb
- V_{DD} =5V → Rss=7kΩ
- 温度 =50°C （系统最大值）
- V_{HOLD} =0V @ time = 0

例 17-1： 计算所需最小采集时间

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$

只有在温度 > 25°C 时，才需要温度系数。

$$T_{ACQ} = 2\ \mu s + T_C + [(Temp - 25°C)(0.05\ \mu s/°C)]$$

$$T_C = -C_{HOLD}\ (R_{IC} + R_{SS} + R_S)\ \ln(1/2048)$$
$$-120\ pF\ (1\ k\Omega + 7\ k\Omega + 2.5\ k\Omega)\ \ln(0.0004883)$$
$$-120\ pF\ (10.5\ k\Omega)\ \ln(0.0004883)$$
$$-1.26\ \mu s\ (-7.6246)$$
$$9.61\ \mu s$$

$$T_{ACQ} = 2\ \mu s + 9.61\ \mu s + [(50°C - 25°C)(0.05\ \mu s/°C)]$$
$$11.61\ \mu s + 1.25\ \mu s$$
$$12.86\ \mu s$$

17.2 选择 A/D 转换时钟

每一位的 A/D 转换时间被定义为 TAD。每完成一次 10 位 A/D 转换需要 12TAD。A/D 转换的时钟源可用软件进行选择。对于 TAD 可以有以下 7 种选择：

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- 内部 A/D 模块 RC 振荡器 （2-6 μs）

为了使 A/D 转换正确， A/D 转换时钟 （TAD）的选择必须确保 TAD 最小值不得低于 1.6 μs。

表 17-1 显示了器件在不同工作频率下以及所选的不同 A/D 时钟源下得到的 TAD。

17.3 配置模拟端口引脚

ADCON1、 TRISA 和 TRISE 寄存器用来控制 A/D 端口引脚的操作。若希望端口引脚为模拟输入，则必须将其相应的 TRIS 位置 1（输入）；如果将 TRIS 位清零（输出），则会转换为数字输出电平 （VOH 或 VOL）。

A/D 转换与 CHS2:CHS0 位及 TRIS 位的状态无关。

注 1：读取端口寄存器时，所有配置为模拟输入通道的引脚均读为 0（低电平）。配置为数字输入的引脚将转换模拟输入信号。配置为数字输入的引脚上的模拟电平不会影响转换精度。

2：定义为数字输入的引脚上的模拟电平（包括 AN4:AN0 引脚），可能导致输入缓冲器消耗的电流超出器件规范。

表 17-1： TAD 与器件工作频率

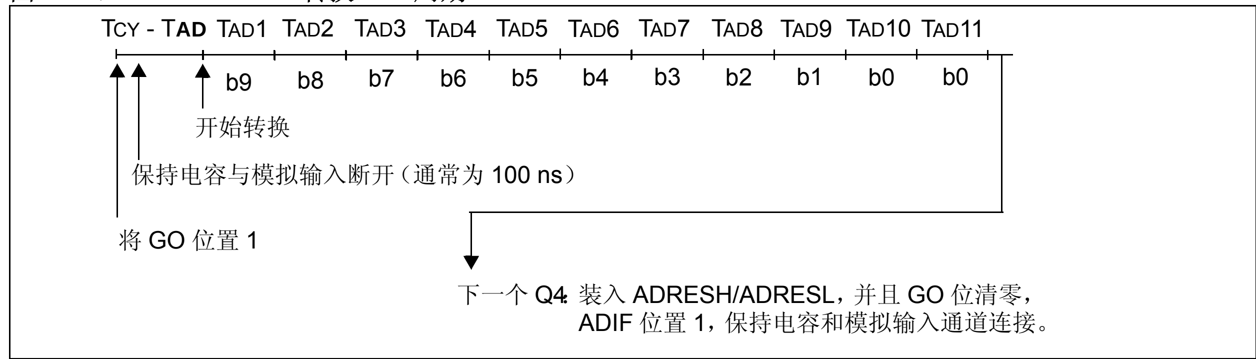
AD 时钟源 （TAD）		最高器件频率	
工作状态	ADCS2:ADCS0	PIC18FXX2	PIC18LFXX2
2 TOSC	000	1.25 MHz	666 kHz
4 TOSC	100	2.50 MHz	1.33 MHz
8 TOSC	001	5.00 MHz	2.67 MHz
16 TOSC	101	10.00 MHz	5.33 MHz
32 TOSC	010	20.00 MHz	10.67 MHz
64 TOSC	110	40.00 MHz	21.33 MHz
RC	011	—	—

17.4 A/D 转换

图 17-3 显示了在 GO 位置 1 后， A/D 转换器的工作状态。在转换期间将 GO/DONE 位清零将中止当前 A/D 转换。部分完成的 A/D 转换样本不会更新 A/D 结果寄存器对。即， ADRESH:ADRESL 寄存器仍然保持上一次转换完成后的结果（即上一次写入 ADRESH:ADRESL 寄存器中的值）。中止 A/D 转换后，需要等待 2TAD 的时间才开始下一次采集。等待 2TAD 之后，将自动开始对所选通道进行采集。可以将 GO/DONE 位置 1，并开始 A/D 转换。

注：	不应在打开 A/D 模块的同一指令中将 GO/DONE 位置 1。

图 17-3： A/D 转换 TAD 周期



17.4.1 A/D 结果寄存器

在 A/D 转换结束后，其 10 位结果会存放于 ADRESH:ADRESL 寄存器对中。这对寄存器有 16 位宽。可以灵活选择该 A/D 模块的 10 位结果是以左对齐还是右对齐的方式存放在 16 位结果寄存器中。 A/D 结果格式选择位 （ADFM）控制对齐方式。图 17-4 显示了 A/D 结果对齐的操作。多余位填入 “0”。当 A/D 结果不改写这些单元（A/D 被禁止）时，这些寄存器可用作两个通用的 8 位寄存器。

图 17-4： A/D 结果的对齐方式

10 位结果
ADFM=1
ADFM=0
7 2 1 0 7 0
0000 00
ADRESH ADRESL
10 位结果
右对齐
7 0 7 6 5 0
0000 00
ADRESH ADRESL
10 位结果
左对齐

17.5 CCP2 触发器的使用

CCP2 模块的 “特殊事件触发器” 可以启动 A/D 转换。要求将 CCP2M3:CCP2M0 位（CCP2CON<3:0>）设置为 1011，且使能 A/D 模块（ADON 位置 1）。发生触发时，GO/DONE 位被置 1，启动 A/D 转换，Timer1（或 Timer3）计数器被复位为 0。复位 Timer1（或 Timer3）可自动重复 A/D 采集周期，最大限度地降低了软件开销（将 ADRESH/ADRESL 移到期望的位置）。

在 “特殊事件触发器” 将 GO/$\overline{\text{DONE}}$ 位置 1（启动转换）前，必须正确选择模拟输入通道，并要保证最小采集时间。

如果未使能 A/D 模块（ADON 清零），则 “特殊事件触发器” 将被 A/D 模块忽略，但它仍会复位 Timer1（或 Timer3）计数器。

表 17-2： A/D 寄存器小结

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	POR 和 BOR 时的值	其他复位时的值
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF[(1)]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE[(1)]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP[(1)]	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
PIR2	—	—	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	---0 0000	---0 0000
PIE2	—	—	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	---0 0000	---0 0000
IPR2	—	—	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	---1 1111	---1 0000
ADRESH	A/D 结果寄存器								xxxx xxxx	uuuu uuuu
ADRESL	A/D 结果寄存器								xxxx xxxx	uuuu uuuu
ADCON0	ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	—	ADON	0000 00-0	0000 00-0
ADCON1	ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0	---- -000	---- -000
PORTA	—	RA6	RA5	RA4	RA3	RA2	RA1	RA0	--0x 0000	--0u 0000
TRISA	—	PORTA 数据方向寄存器							--11 1111	--11 1111
PORTE	—	—	—	—	—	RE2	RE1	RE0	---- -000	---- -000
LATE	—	—	—	—	—	LATE2	LATE1	LATE0	---- -xxx	---- -uuu
TRISE	IBF	OBF	IBOV	PSPMODE	—	PORTE 数据方向位			0000 -111	0000 -111

图注： x = 未知， u = 不变， - = 未实现，读作 0。阴影单元不适用于 A/D 转换。

注 1： PIC18F2X2 器件的 PSPIF、 PSPIE 和 PSPIP 位是保留位，其值始终为 0。

18.0 低压检测

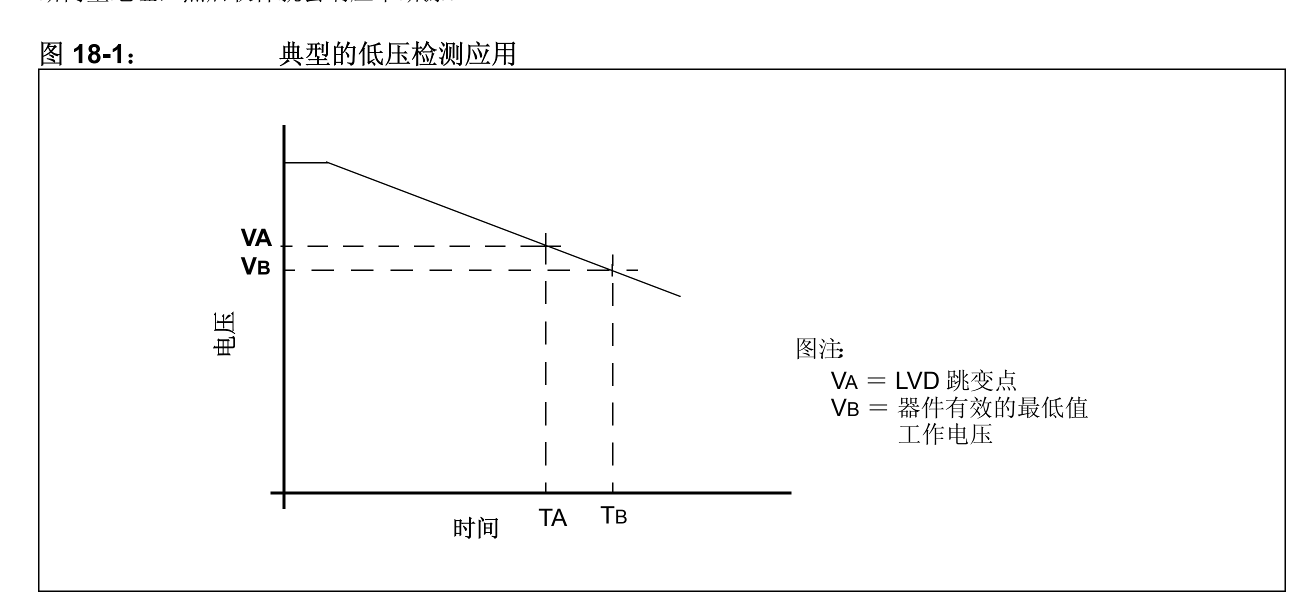
在很多应用场合都希望能够确定器件电压（V_{DD}）是否低于指定电压。可以为该应用建立一个工作窗口，在器件工作电压偏离有效范围之前，应用软件在该工作窗口中可进行一些“保护性工作”。可以使用低压检测模块进行判断。

这一模块是一个可用软件编程的电路，可以用来指定器件的电压跳变点。当器件电压低于指定值时，对中断标志进行置 1。如果使能了中断，程序执行时会转移到中断向量地址，然后软件就会响应中断源。

低压检测电路完全由软件控制。这样就允许通过软件“关断”电路，使器件的电流消耗最小。

图 18-1 所示是一种可能的应用的电压曲线（通常适用于电池）。器件电压会随时间而逐渐下降。当器件电压等于电压 V_A 时，LVD 逻辑会发出一个中断。这一情况发生在 T_A 时刻。于是应用软件有足够的时间去关闭系统，直到器件电压超出有效工作范围为止。电压点 V_B 是最低有效工作电压的指定值。这一情况发生在 T_B 时刻。（T_B - T_A）的差就是总的关断时间。

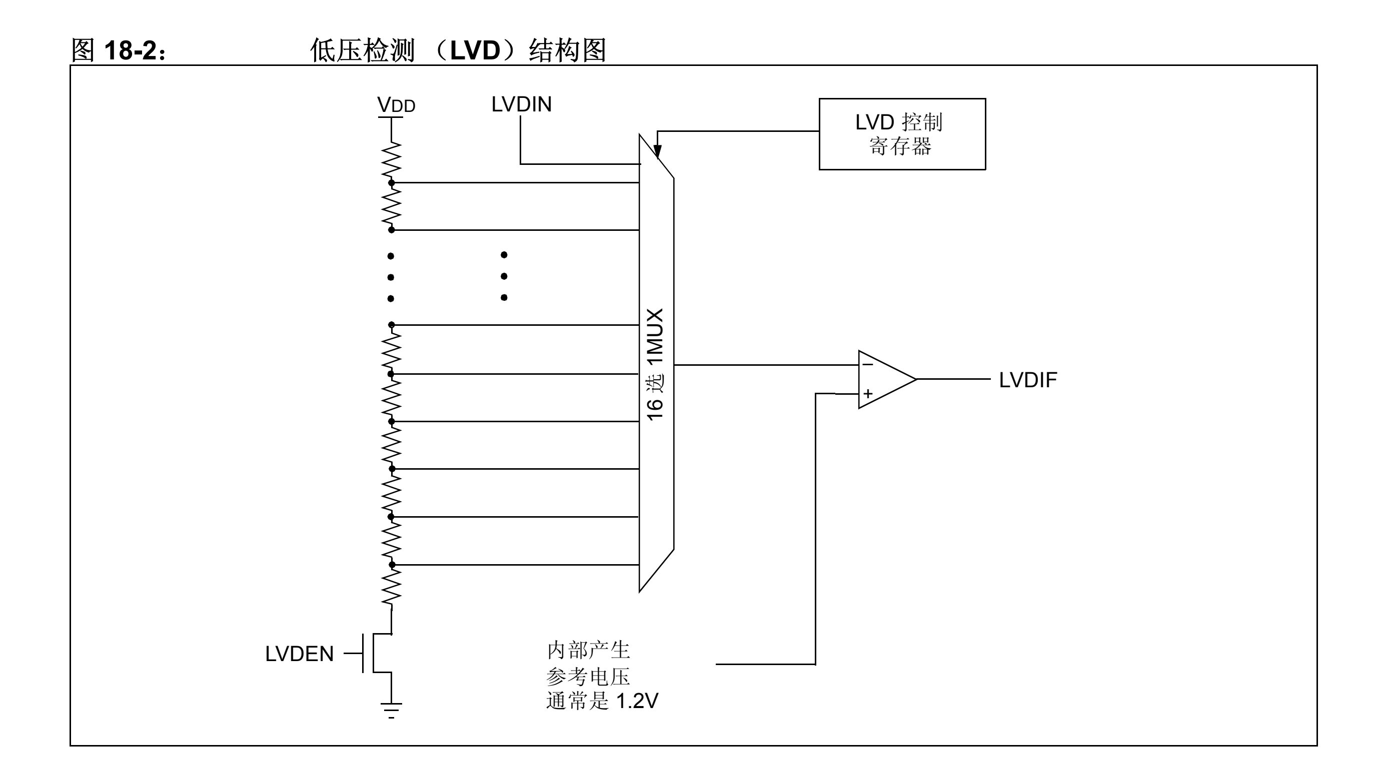
图 18-1： 典型的低压检测应用



LVD 模块的结构图如图 18-2 所示。比较器使用内部产生的参考电压作为设定点。当器件电压在选定触点上的输出越过了（低于）设定点，则将 LVDIF 位置 1。

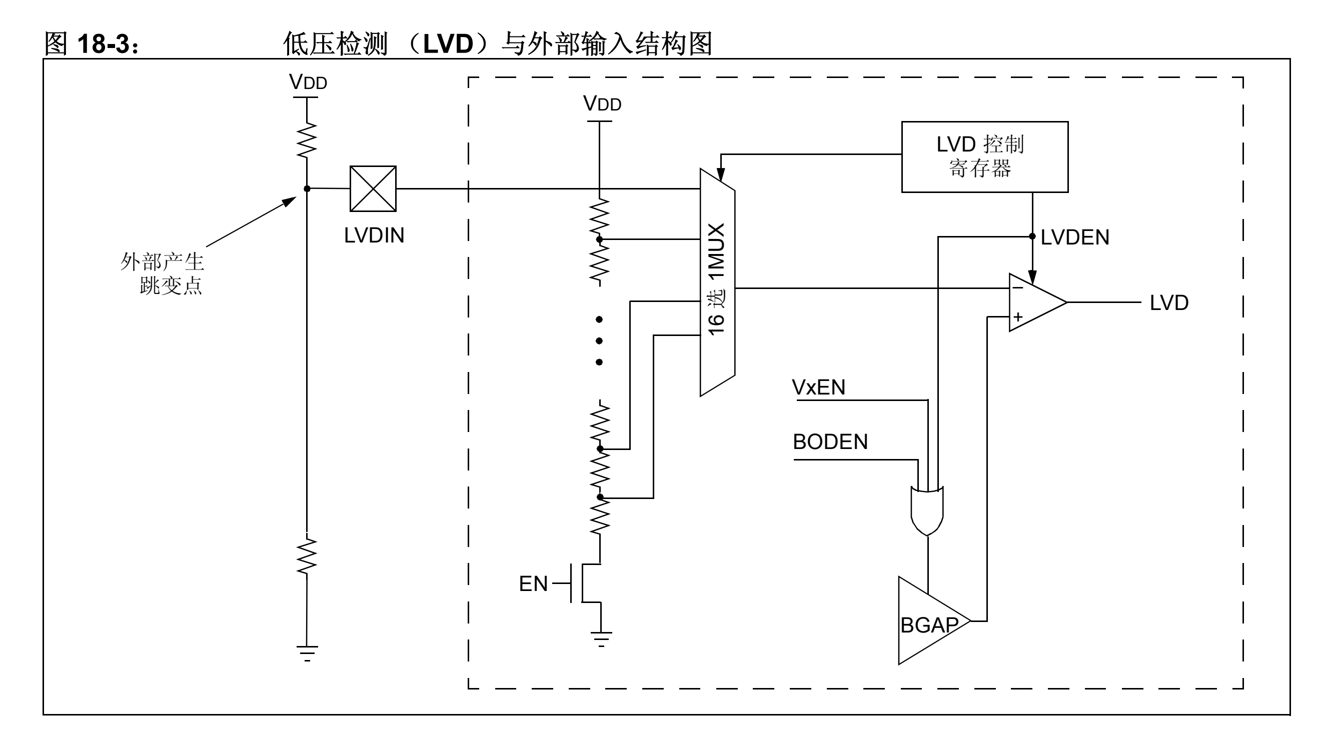
电阻分压器的每个节点代表一个“跳变点”电压。该“跳变点”电压是在 LVD 模块发出中断前器件工作所需的最低电压。当电压等于跳变点时，从电阻阵列分出的电压等于 1.2V 内部参考电压，这一电压由电压参考模块产生。然后比较器发出一个中断信号将 LVDIF 位置 1。这个电压可以用软件编程控制，共有 16 个值可供选择（见图 18-2）。通过程序控制 LVDL3:LVDL0 位（LVDCON<3:0>）来选择跳变点。

图 18-2： 低压检测 （LVD）结构图



LVD 模块还有一个功能，可以让用户从外部源为模块提供跳变电压。当 LVDL3:LVDL0 位被设置为 1111 时，使能这一模式。在这种状态下，比较器输入与外部输入引脚 LVDIN 复用 （图 18-3）。这给了用户极大的灵活性，他们可以配置在有效工作范围内的任何电压下发生低压检测中断。

图 18-3： 低压检测 （LVD）与外部输入结构图



18.1 控制寄存器

低压检测控制寄存器控制低压检测电路的工作。

寄存器 18-1： LVDCON 寄存器

U-0	U-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0
bit 7							bit 0

bit 7-6 **未实现：**读作 0

bit 5 **IRVST：**内部参考电压稳定标志位
1 ＝表明低压检测逻辑将会在特定电压范围产生中断标志
0 ＝表明低压检测逻辑将不会在特定电压范围产生中断标志，不要使能 LVD 中断

bit 4 **LVDEN：**低压检测电源使能位
1 ＝ 使能 LVD，给 LVD 电路上电
0 ＝ 禁止 LVD， LVD 电路掉电

bit 3-0 **LVDL3:LVDL0：**低压检测限制位
1111 ＝使用外部模拟输入 （输入来自于 LVDIN 引脚）
1110 ＝ 4.5V - 4.77V
1101 ＝ 4.2V - 4.45V
1100 ＝ 4.0V -4.24V
1011 ＝ 3.8V -4.03V
1010 ＝ 3.6V -3.82V
1001 ＝ 3.5V -3.71V
1000 ＝ 3.3V -3.50V
0111 ＝ 3.0V - 3.18V
0110 ＝ 2.8V - 2.97V
0101 ＝ 2.7V - 2.86V
0100 ＝ 2.5V - 2.65V
0011 ＝ 2.4V - 2.54V
0010 ＝ 2.2V - 2.33V
0001 ＝ 2.0V - 2.12V
0000 ＝ 保留

注： 没有测试 LVDL3:LVDL0 模式下会出现的低于器件有效工作电压的跳变点。

图注：			
R ＝可读位	R ＝可写位	U ＝未实现位，读作 0	
- n ＝ POR 时的值	1 ＝ 置位	0 ＝ 清零	x ＝未知

18.2 工作方式

通常电压下降相对较慢，这取决于器件电压的电源。这意味着 LVD 模块不需要经常工作。为了降低电流要求，LVD 电路只需要在检测电压的短暂时间内使能。检测完成之后，可以禁止 LVD 模块。

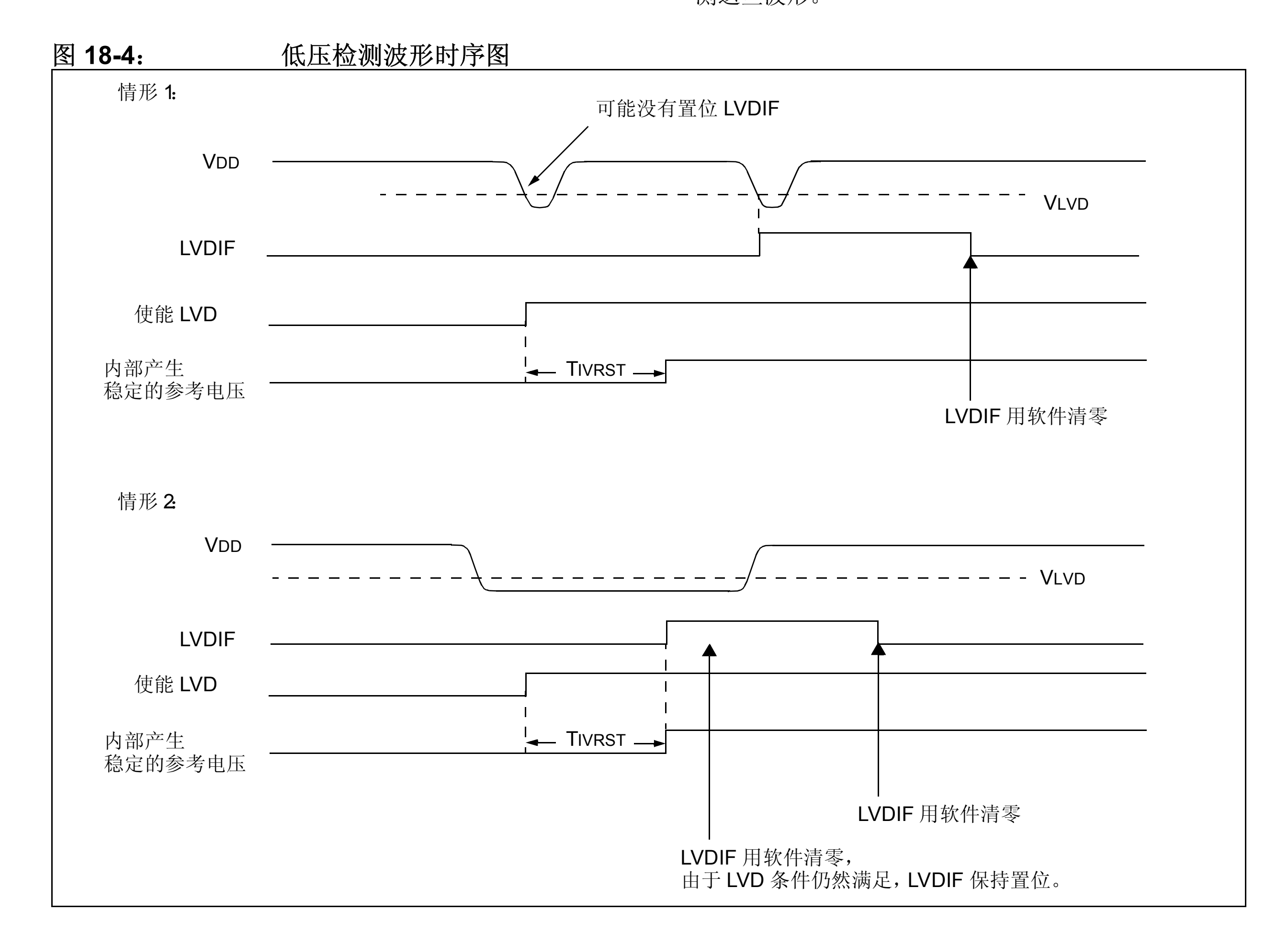
每次使能 LVD 模块时，电路需要过些时间才能稳定下来。当电路稳定之后，可以将所有状态标志都清零。然后，将指明系统的正确状态。

设置 LVD 模块的步骤如下：

1. 将表示所选 LVD 跳变点的值写入 LVDL3:LVDL0 位（LVDCON 寄存器）。
2. 确保 LVD 中断被禁止（LVDIE 位或 GIE 位被清零）。
3. 使能 LVD 模块（将 LVDCON 寄存器中的 LVDEN 位置 1）。
4. 等待 LVD 模块稳定下来（当 IRVST 位置 1）。
5. 由于 LVD 中断标志位在 LVD 模块稳定前可能被错误地置 1，将该标志位清零（LVDIF 位清零）。
6. 使能 LVD 中断（LVDIE 位和 GIE 位置 1）。

图 18-4 是一些典型的波形时序图，用 LVD 模块可以检测这些波形。

图 18-4： 低压检测波形时序图



18.2.1 参考电压设定点

LVD 模块的内部参考电压可以供其他内部电路使用（可编程的欠压复位）。如果禁止这些电路（降低电流消耗），参考电压电路需要一段时间才能稳定下来，之后才能可靠地对低压情况进行检测。这一时间不随系统时钟速率变化。启动时间在电气规格参数 36 中标出。只有在到达稳定的参考电压以后，才使能低压中断标志。请参考图 18-4 中的波形时序图。

18.2.2 电流消耗

如果模块被使能，LVD 比较器和分压器就会使能，并消耗静态电流。分压器可以从电阻阵列中的多处接出。在使能情况下的电流消耗总量在电气规格参数 #D022B 中标出。

18.3 休眠时的工作

如果使能的话，LVD 电路在休眠期间将继续工作。如果器件电压越过了跳变点， LVDIF 位将会被置 1，并且从休眠状态中唤醒器件。如果中断已被全局使能，器件会从中断向量地址处继续执行。

18.4 复位的影响

器件复位会迫使所有寄存器进入复位状态。这会导致 LVD 模块被关断。

注：

19.0 CPU 特殊功能

有些功能旨在最大限度提高系统可靠性，通过删除外部组件将成本降至最低，提供节能操作模式以及代码保护功能。它们是：

- OSC 选择
- 复位
  - 上电复位 （POR）
  - 上电延时定时器 （PWRT）
  - 振荡器起振定时器 （OST）
  - 欠压复位 （BOR）
- 中断
- 看门狗定时器 （WDT）
- 休眠
- 代码保护
- ID 单元
- 在线串行编程

所有的 PIC18FXX2 器件都有一个看门狗定时器，它可以通过设置配置位或软件控制实现永久使能。它依靠自带的 RC 振荡器来增加可靠性。有两个定时器提供必要的上电延时。一个是振荡器起振定时器 （OST），确保在晶体振荡器的振荡达到稳定前，器件处于复位状态。另外一个是上电延迟定时器 （PWRT），仅提供一个固定的上电延时，使器件在电源稳定前保持复位状态。有这两个片内定时器，大部分的应用无需外接复位电路。

休眠模式用来提供一个电流极低的掉电模式。用户可通过外部复位、看门狗定时器唤醒或中断将器件从休眠状态唤醒。也可使用某些振荡器模式，使得器件更适合应用要求。选择 RC 振荡器模式可节省系统的成本，而 LP 晶振模式可降低系统功耗。可以使用一组配置位来选择不同的模式。

19.1 配置位

可以通过对配置位编程 （读为 0）或不编程 （读作 1）来选择不同的器件配置。这些配置位从程序存储器单元 300000h 开始映射。

用户会注意到地址300000h超出了用户程序存储器空间范围。事实上，它属于配置存储器空间 （300000h － 3FFFFFh），仅能通过读表和写表来存取。

对配置寄存器编程类似于对闪存存储器编程 （参见第 5.5.1节）。唯一区别在于，每次往配置寄存器写入一个字节。对配置寄存器进行编程的事件顺序如下：

1. 将要写入的配置寄存器地址载入表指针。
2. 使用 `TBLWT` 指令写入一个字节。
3. 使 EEPGD 指向程序存储器，将 CFGS 位置 1 以访问配置寄存器，将 WREN 位置 1 以使能字节写入。
4. 禁止中断。
5. EECON2 内写入 55h。
6. EECON2 内写入 AAh。
7. 将 WR 位置 1。开始写周期。
8. CPU 在写操作期间将停止运行 （在使用内部定时器时大约 2 ms）。
9. 执行一条 `NOP` 指令。
10. 重新使能中断。

表 19-1： 配置位和器件 ID

文件名称		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	缺省值 / 未编程值
300001h	CONFIG1H	—	—	$\overline{\text{OSCSEN}}$	—	—	FOSC2	FOSC1	FOSC0	--1- -111
300002h	CONFIG2L	—	—	—	—	BORV1	BORV0	BOREN	$\overline{\text{PWRTEN}}$	---- 1111
300003h	CONFIG2H	—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEN	---- 1111
300005h	CONFIG3H	—	—	—	—	—	—	—	CCP2MX	---- ---1
300006h	CONFIG4L	$\overline{\text{DEBUG}}$	—	—	—	—	LVP	—	STVREN	1--- -1-1
300008h	CONFIG5L	—	—	—	—	CP3	CP2	CP1	CP0	---- 1111
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	11-- ----
30000Ah	CONFIG6L	—	—	—	—	WRT3	WRT2	WRT1	WRT0	---- 1111
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	111- ----
30000Ch	CONFIG7L	—	—	—	—	EBTR3	EBTR2	EBTR1	EBTR0	---- 1111
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	-1-- ----
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	**(1)**
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	0000 0100

图注： x= 未知， u= 不变， -= 未实现， q= 取值根据情况而定。
阴影单元未实现，读为 0。

注 **1**：关于 DEVID1 的值，请参见寄存器 19-12。

寄存器 19-1： 配置寄存器 1 高位 （CONFIG1H：字节地址 300001h）

U-0	U-0	R/P-1	U-0	U-0	R/P-1	R/P-1	R/P-1
—	—	$\overline{\text{OSCSEN}}$	—	—	FOSC2	FOSC1	FOSC0
bit 7							bit 0

bit7-6 **未实现**：读作 0

bit5 **$\overline{\text{OSCSEN}}$**：振荡器系统时钟开关使能位
1= 振荡器系统时钟切换选项被禁止 （主振荡器作为时钟源）
0= 振荡器系统时钟切换选项使能 （振荡器切换使能）

bit4-3 **未实现**：读为 0

bit2-0 **FOSC2:FOSC0**：振荡器选择位
111=RC 振荡器， OSC2 被配置为 RA6
110=PLL 使能的 HS 振荡器 / 时钟频率 =(4 × FOSC)
101=EC 振荡器， OSC2 被配置为 RA6
100=EC 振荡器， OSC2 被配置为四分频时钟输出
011=RC 振荡器
010=HS 振荡器
001=XT 振荡器
000=LP 振荡器

图注：		
R= 可读位	P= 可编程位	U= 未实现位，读作 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-2： 配置寄存器 2 低位 （CONFIG2L：字节地址 300002h）

U-0	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	BORV1	BORV0	BOREN	$\overline{\text{PWRTEN}}$
bit 7							bit 0

bit7-4 **未实现**：读作 0

bit3-2 **BORV1:BORV0**：欠压复位电压位
11=V_{BOR} 置为 2.5V
10=V_{BOR} 置为 2.7V
01=V_{BOR} 置为 4.2V
00=V_{BOR} 置为 4.5V

bit1 **BOREN**：欠压复位使能位
1= 使能欠压复位
0= 禁止欠压复位

bit0 **$\overline{\text{PWRTEN}}$**：上电延时定时器使能位
1= 禁止 PWRT
0= 使能 PWRT

图注：		
R= 可读位	P= 可编程位	U= 未实现位，读作 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-3： 配置寄存器 2 高位 （CONFIG2H：字节地址 300003h）

U-0	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	WDTPS2	WDTPS1	WDTPS0	WDTEN
bit 7							bit 0

bit7-4 **未实现**：读作 0

bit3-1 **WDTPS2:WDTPS0**：看门狗定时器后分频选择位
111 = 1:128
110 = 1:64
101 = 1:32
100 = 1:16
011 = 1:8
010 = 1:4
001 = 1:2
000 = 1:1

bit0 **WDTEN**：看门狗定时器使能位
1= 使能 WDT
0= 禁止 WDT （控制位为 SWDTEN 位）

图注：		
R= 可读位	P= 可编程位	U= 未实现位，读作 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-4： 配置寄存器 3 高位 （**CONFIG3H**：字节地址 **300005h**）

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/P-1
—	—	—	—	—	—	—	CCP2MX
bit 7							bit 0

bit7-1 未实现：读作 0

bit0 **CCP2MX**：CCP2 复用位
1=CCP2 输入 / 输出与 RC1 复用
0=CCP2 输入 / 输出与 RB3 复用

图注：		
R= 可读位	P= 可编程位	U= 未实现位，读为 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-5： 配置寄存器 4 低位 （**CONFIG4L**：字节地址 **300006h**）

R/P-1	U-0	U-0	U-0	U-0	R/P-1	U-0	R/P-1
$\overline{\text{BKBUG}}$	—	—	—	—	LVP	—	STVREN
bit 7							bit 0

bit7 **$\overline{\text{DEBUG}}$**：后台调试器使能位
1= 禁止后台调试器。 RB6 和 RB7 配置为通用的 I/O 引脚。
0= 使能后台调试器。 RB6 和 RB7 专用于在线调试。

bit6-3 未实现：读为 0

bit2 **LVP**：低电压 ICSP 使能位
1= 使能低电压 ICSP
0= 禁止低电压 ICSP

bit1 未实现：读为 0

bit0 **STVREN:** 堆栈满 / 下溢复位使能位
1= 堆栈满 / 下溢会导致复位
0= 堆栈满 / 下溢不会导致复位

图注：		
R= 可读位	C= 可清零位	U= 未实现位，读为 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-6： 配置寄存器 5 低位 （CONFIG5L：字节地址 300008h）

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	CP3[1]	CP2[1]	CP1	CP0
bit 7							bit 0

bit7-4 **未实现**：读为 0

bit3 **CP3**：代码保护位 [1]
1= 地址块 3 （006000-007FFFh）无代码保护
0= 地址块 3 （006000-007FFFh）有代码保护

bit2 **CP2**：代码保护位 [1]
1= 地址块 2 （004000-005FFFh）无代码保护
0= 地址块 2 （004000-005FFFh）有代码保护

bit1 **CP1**：代码保护位
1= 地址块 1 （002000-003FFFh）无代码保护
0= 地址块 1 （002000-003FFFh）有代码保护

bit0 **CP0**：代码保护位
1= 地址块 0 （000200-001FFFh）无代码保护
0= 地址块 0 （000200-001FFFh）有代码保护

注 **1**：在 PIC18FX42 器件中未实现；保持该位置 1。

图注：		
R= 可读位	C= 可清零位	U= 未实现位，读为 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-7： 配置寄存器 5 高位 （CONFIG5H：字节地址 300009h）

R/C-1	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
CPD	CPB	—	—	—	—	—	—
bit 7							bit 0

bit7 **CPD**：数据 EEPROM 代码保护位
1= 数据 EEPROM 无代码保护
0= 数据 EEPROM 有代码保护

bit6 **CPB**：引导地址块代码保护位
1= 引导地址块 （000000-001FFFh）无代码保护
0= 引导地址块 （000000-001FFFh）有代码保护

bit5-0 **未实现**：读为 0

图注：		
R= 可读位	C= 可清零位	U= 未实现位，读为 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-8：　配置寄存器低 6 位 （CONFIG6L：字节地址 30000Ah）

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	WRT3[(1)]	WRT2[(1)]	WRT1	WRT0
bit 7							bit 0

bit7-4　**未实现**：读为 0

bit3　**WRT3**：写保护位 [(1)]
1= 地址块 3 （006000-007FFFh）无写保护
0= 地址块 3 （006000-007FFFh）有写保护

bit2　**WRT2**：写保护位 [(1)]
1= 地址块 2 （004000-005FFFh）无写保护
0= 地址块 2 （004000-005FFFh）有写保护

bit1　**WRT1**：写保护位
1= 地址块 1 （002000-003FFFh）无写保护
0= 地址块 1 （002000-003FFFh）有写保护

bit0　**WRT0**：写保护位
1= 地址块 0 （000200h-001FFFh）无写保护
0= 地址块 0 （000200h-001FFFh）有写保护

注　**1**：PIC18FX42 器件中未实现；保持该位置 1。

图注：		
R= 可读位	C= 可清零位	U= 未实现位，读为 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-9：　配置寄存器 6 高位 （CONFIG6H：字节地址 30,000Bh）

R/C-1	R/C-1	C-1	U-0	U-0	U-0	U-0	U-0
WRTD	WRTB	WRTC	—	—	—	—	—
bit 7							bit 0

bit7　**WRTD**：数据 EEPROM 写保护位
1= 数据 EEPROM 无写保护
0= 数据 EEPROM 有写保护

bit6　**WRTB**：引导模块写保护位
1= 引导地址块 （000000-001FFFh）无写保护
0= 引导地址块 （000000-001FFFh）有写保护

bit5　**WRTC**：配置寄存器写保护位
1= 配置寄存器 （300000-3000FFh）无写保护
0= 配置寄存器 （300000-3000FFh）有写保护
注：　该位只读，并且不能在用户模式下更改。

bit4-0　**未实现**：读为 0

图注：		
R= 可读位	C= 可清零位	U= 未实现位，读为 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-10： 配置寄存器 7 低位 （CONFIG7L：字节地址 30000Ch）

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	EBTR3[1]	EBTR2[1]	EBTR1	EBTR0
bit 7							bit 0

bit7-4 **未实现**：读为 0

bit3 **EBTR3**：读表保护位[1]
1= 地址块 3 （006000-007FFFh）未采用读保护，可从其他地址块对其执行读表操作
0= 地址块 3 （006000-007FFFh）采用读保护，不能从其他地址块对其执行读表操作

bit2 **EBTR2**：读表保护位[1]
1= 地址块 2 （004000-005FFFh）未采用读保护，可从其他地址块对其执行读表操作
0= 地址块 2 （004000-005FFFh）采用读保护，不能从其他地址块对其执行读表操作

bit1 **EBTR1**：读表保护位
1= 地址块 1 （002000-003FFFh）未采用读保护，可从其他地址块对其执行读表操作
0= 地址块 1 （002000-003FFFh）采用读保护，不能从其他地址块对其执行读表操作

bit0 **EBTR0**：读表保护位
1= 地址块 0 （000200h-001FFFh）未采用读保护，可从其他地址块对其执行读表操作
0= 地址块 0 （000200h-001FFFh）采用读保护，不能从其他地址块对其执行读表操作

注 **1**：PIC18FX42 器件中未实现；保持该位置 1。

图注：		
R= 可读位	C= 可清零位	U= 未实现位，读为 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-11： 配置寄存器 7 高位 （CONFIG7H：字节地址 30000Dh）

U-0	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
—	EBTRB	—	—	—	—	—	—
bit 7							bit 0

bit7 **未实现**：读为 0

bit6 **EBTRB**：引导地址块读表保护位
1= 引导地址块 （000000-005FFFh）未采用读保护，可从其他地址块对其执行读表操作
0= 引导地址块 （000000-005FFFh）采用读保护，不能从其他地址块对其执行读表操作

bit5-0 **未实现**：读为 0

图注：		
R= 可读位	C= 可清零位	U= 未实现位，读为 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-12： PIC18FXX2 的器件 ID 寄存器 1 （DEVID1：字节地址 3FFFFEh）

R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0
bit 7							bit 0

bit7-5 **DEV2:DEV0**：器件 ID 位
000=PIC18F252
001=PIC18F452
100=PIC18F242
101=PIC18F442

bit4-0 **REV4:REV0**：版本 ID 位
这些位用于表明器件版本。

图注：		
R= 可读位	P= 可编程位	U= 未实现位，读为 0
-n= 器件未编程时的值		u= 编程后状态不变

寄存器 19-13： PIC18FXX2 的器件 ID 寄存器 2 （DEVID2：字节地址 3FFFFFh）

R	R	R	R	R	R	R	R
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3
bit 7							bit 0

bit7-0 **DEV10:DEV3**：器件 ID 位
这些位与器件 ID 寄存器 1 中的 DEV2:DEV0 位结合使用以确定部件号。

图注：		
R= 可读位	P= 可编程位	U= 未实现位，读为 0
-n= 器件未编程时的值		u= 编程后状态不变

19.2 看门狗定时器 （WDT）

看门狗定时器是一个片内自由运行的 RC 振荡器，它不需要任何的外接元件。该 RC 振荡器独立于 OSC1/CLKIN 引脚上的 RC 振荡器。这样，即使器件的 OSC1/CLK1 和 OSC2/CLKO/RA6 引脚上的时钟停止（例如执行了 SLEEP 指令）， WDT 仍将正常工作。

在正常工作期间， WDT 超时将导致器件复位 （看门狗定时器复位）。如果器件处于休眠状态，则 WDT 超时将唤醒器件，使其继续正常工作（看门狗定时器唤醒）。RCON 寄存器中 $\overline{\text{TO}}$ 位将在 WDT 超时时被清零。

看门狗定时器通过一个器件配置位被使能 / 禁止。如果 WDT 被使能，就不能通过软件禁止此功能。当 WDTEN 配置位被清零，SWDTEN 位控制 WDT 操作的使能 / 禁止。

WDT 超时值可以在电气特性（第 22.0 节）的参数 D031 下找到。可以通过配置位对 WDT 的后分频器值进行赋值。

> **注：** `CLRWDT` 和 `SLEEP` 指令将 WDT 和后分频器（如果分配给 WDT）清零，防止其超时而引起器件复位。

> **注：** 当执行一个 `CLRWDT` 指令，且后分频器被分配给 WDT 时，后分频器的计数将清零，但是它的赋值不会改变。

19.2.1 控制寄存器

寄存器 19-14 表示 WDTCON 寄存器。这是一个可读写寄存器，它包含一个控制位，该控制位允许软件仅在 WDT 使能配置位禁止 WDT 时改写该配置位。

寄存器 19-14： WDTCON 寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SWDTEN
bit 7							bit 0

bit7-1 **未实现**：读为 0

bit0 **SWDTEN**：软件控制的看门狗定时器使能位
1= 打开看门狗定时器
0= 如果配置寄存器中的 WDTEN 配置位为 0，
则关闭看门狗定时器。

图注：	
R= 可读位	W= 可写位
U= 未实现位，读为 0	-n= 上电复位值

19.2.2 WDT 后分频器

WDT 的后分频器可以延长 WDT 复位时间段。在器件编程时，根据写入 CONFIG2H 配置寄存器的值来选中后分频器。

图 19-1： 看门狗定时器结构框图

WDT 定时器
后分频器
8
8 选 1 多路开关
WDTPS2:WDTPS0
WDTEN
配置位
SWDTEN 位
WDT
超时

注： WDPS2:WDPS0 是 CONFIG2H 寄存器中的位。

表 19-2： 看门狗定时器寄存器小结

名称	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG2H	—	—	—	—	WDTPS2	WDTPS2	WDTPS0	WDTEN
RCON	IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
WDTCON	—	—	—	—	—	—	—	SWDTEN

图注： 阴影单元不适用于看门狗定时器。

19.3 掉电模式（休眠）

通过执行 SLEEP 指令进入掉电模式。

如果使能掉电模式，看门狗定时器将被清零，但是保持运行，$\overline{\text{PD}}$ 位（RCON<3>）被清零，$\overline{\text{TO}}$ 位（RCON<4>）被置 1，并且振荡器分频器被关闭。 I/O 端口保持执行 SLEEP 指令前的状态（驱动为高电平、低电平或高阻抗）。

要使这种模式下电流消耗最小，把所有的 I/O 引脚都接到 V_{DD} 或 V_{SS}，并确保没有外部电路从该 I/O 引脚分流，同时关闭 A/D 转换器，禁用外部时钟。为了避免输入端悬空引起开关电流，应从外部拉高或拉低所有高阻抗输入的I/O引脚。将T0CKI的输入设置为V_{DD}或V_{SS}，使电流消耗最小。还要考虑到 PORTB 上的片内上拉造成的影响。

$\overline{\text{MCLR}}$ 引脚必须为逻辑高电平（V_{IHMC}）。

19.3.1 从休眠状态唤醒

通过下列事件之一可唤醒器件：

1. 来自 $\overline{\text{MCLR}}$ 引脚的外部复位输入信号。
2. 看门狗定时器唤醒（如果 WDT 被使能）。
3. 因 INT 引脚、 RB 端口电平变化的中断或外设中断信号。

下列外设中断可以将器件从休眠状态唤醒。

1. PSP 读或写。
2. TMR1 中断。 Timer1 必须用作异步计数器。
3. TMR3 中断。 Timer3 必须用作异步计数器。
4. CCP 捕捉模式中断。
5. 特殊事件触发（Timer1 工作在使用外部时钟的异步模式下）。
6. MSSP（START/STOP）位检测中断。
7. MSSP 在从动模式下发送或接收（SPI/I^2C）。
8. USART RX 或 TX（同步从动模式）。
9. A/D 转换（当 A/D 时钟源是 RC 模式）。
10. EEPROM 写操作完成。
11. LVD 中断。

其他外设不能产生中断，因为在休眠期间没有片内时钟在工作。

外部 $\overline{\text{MCLR}}$ 复位将会导致器件的复位。所有其他事件都被认为是程序执行的后续步骤，将引起"唤醒"。RCON 寄存器中的 $\overline{\text{TO}}$ 和 $\overline{\text{PD}}$ 位可以用来确定引起器件复位的原因。$\overline{\text{PD}}$ 位在上电时被置 1，当执行 SLEEP 时被清零。如果发生 WDT 超时（导致唤醒）， $\overline{\text{TO}}$ 位被清零。

在执行 SLEEP 指令时，下一条指令（PC+2）被预先取出。如果希望通过中断事件唤醒器件，则必须将相应的中断使能位置 1（使能）。唤醒与 GIE 位的状态无关。如果 GIE 位清零（禁止），器件将继续执行 SLEEP 后面的指令。如果 GIE 位置 1（使能），则器件在执行 SLEEP 之后的一条指令后，将跳转到中断地址。如果不希望执行 SLEEP 指令后的指令，应该在 SLEEP 指令后面加一个 NOP 指令。

19.3.2 中断唤醒

当禁止全局中断（GIE 清零）时，且任何中断源的中断使能位和中断标志位均置1时，将会发生下列事件之一：

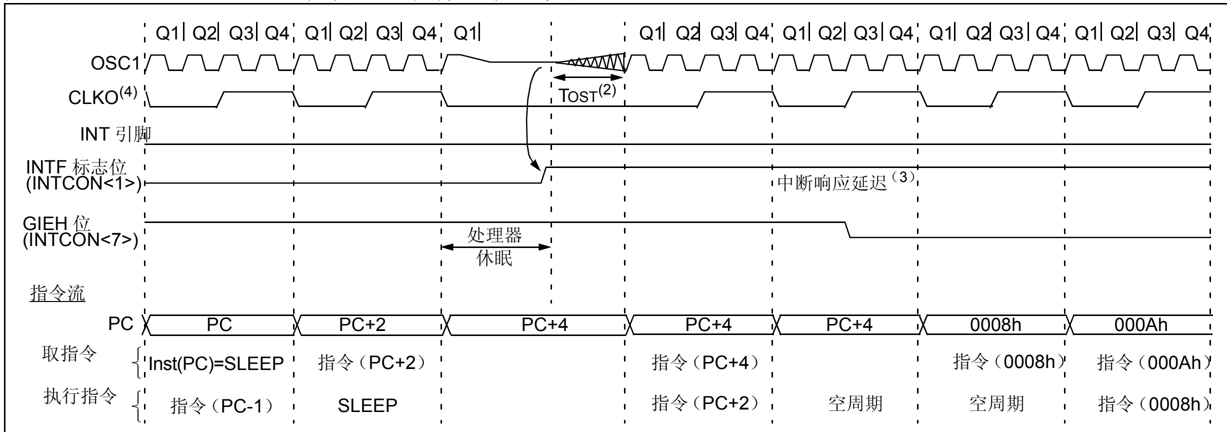
- 如果某个中断（中断标志位和中断使能位都被置 1）在执行 SLEEP 指令**之前**发生， SLEEP 指令将作为 NOP 结束。因此 WDT 和 WDT 后分频器不会被清零， $\overline{\text{TO}}$ 位不会被置 1，而 $\overline{\text{PD}}$ 位也不会被清零。
- 如果在 SLEEP 指令**执行时或执行后**发生中断，器件将立即从休眠状态被唤醒。 SLEEP 指令将在唤醒之前完成执行。因此 WDT 和 WDT 后分频器将被清零， $\overline{\text{TO}}$ 位被置 1，而 $\overline{\text{PD}}$ 位被清零。

即使执行SLEEP指令之前检验了标志位，这些位也可能在 SLEEP 指令完成之前被置 1。要确定是否执行了 SLEEP 指令，可以测试 $\overline{\text{PD}}$ 位。如果 $\overline{\text{PD}}$ 位被置 1，说明 SLEEP 指令作为一条 NOP 指令来执行。

在 SLEEP 指令之前，应先执行一条 CLRWDT 指令，来确保 WDT 被清零。



图 19-2： 通过中断 (1,2) 从休眠状态唤醒



注 1： 假设工作在 XT、 HS 或 LP 振荡器模式。
2： 假设 GIE = '1' 。在这种情况下，处理器被唤醒后，将跳至中断程序。如果 GIE='0'，将在这一行继续执行。
3： TOST=1024TOSC （未按比例画出）。 RC 和 EC 振荡器模式下不会发生这种延时。
4： 在这些振荡器模式下，忽略 CLKO 信号，在这里仅作为时间基准显示出来。

19.4 程序校验和代码保护

PIC18闪存器件的整个代码保护结构与其他的PICmicro系列器件有显著不同。

用户程序存储器分为五个地址块。其中的引导地址块有512 字节。存储器的剩余部分根据二进制边界被分为四个地址块。

这五个地址块中的每一个都有三位代码保护位与之相关联。它们是：

- 代码保护位 （CPn）
- 写保护位 （WRTn）
- 外部地址块读表位 （EBTRn）

图 19-3表示了16KB和32KB器件的程序存储器组织结构，以及与每个地址块相关的具体代码保护位。表 19-3总结了这些位的实际单元。

图 19-3：　受代码保护的程序存储器 PIC18F2XX/4XX

存储器大小 / 器件 16 KB（PIC18FX42）	32 KB（PIC18FX52）	地址范围	地址块代码保护控制位：
引导地址块	引导地址块	000000h 0001FFh	CPB、 WRTB、 EBTRB
地址块 0	地址块 0	000200h 001FFFh	CP0、 WRT0、 EBTR0
地址块 1	地址块 1	002000h 003FFFh	CP1、 WRT1、 EBTR1
未实现的 读为 0	地址块 2	004000h 005FFFh	CP2、 WRT2、 EBTR2
未实现的 读为 0	地址块 3	006000h 007FFFh	CP3、 WRT3、 EBTR3
未实现的 读为 0	未实现的 读为 0	008000h 1FFFFFh	（未实现的存储器空间）

表 19-3：　代码保护寄存器小结

文件名称		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
300008h	CONFIG5L	—	—	—	—	CP3	CP2	CP1	CP0
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—
30000Ah	CONFIG6L	—	—	—	—	WRT3	WRT2	WRT1	WRT0
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—
30000Ch	CONFIG7L	—	—	—	—	EBTR3	EBTR2	EBTR1	EBTR0
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—

图注：　阴影单元未实现。

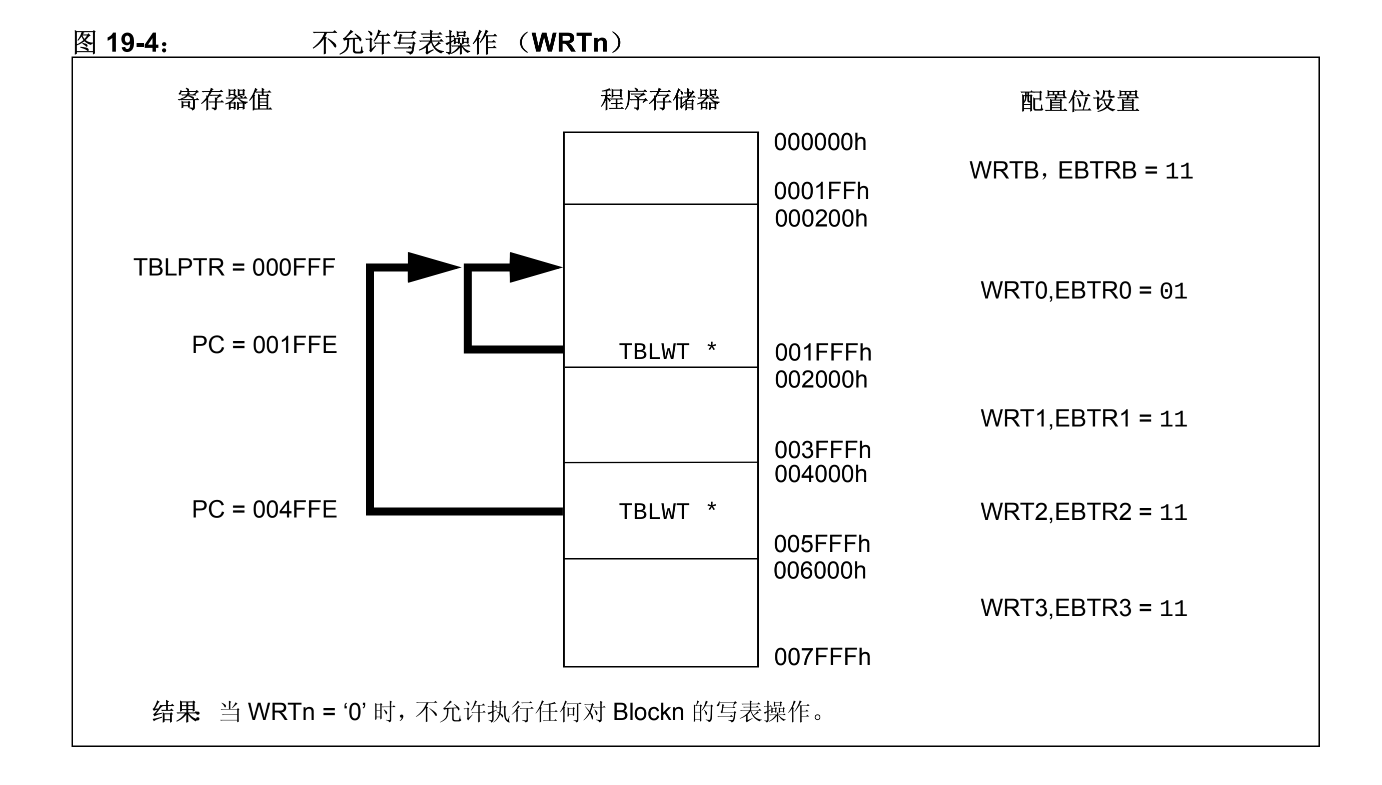
19.4.1 程序存储器代码保护

可以通过读 / 写表指令对用户存储器的任何单元进行读写操作。读表指令会读取器件 ID。读写表指令还可以对配置寄存器进行读写操作。

在用户模式下，CPn 位不产生直接的作用。CPn 位禁止外部的读写操作。如果 WRTn 配置位是 0，可以保护用户存储器地址块不受写表指令的影响。EBTRn位控制读表操作。如果用户存储器某一地址块的 EBTRn 位置为 0，该地址块内的读表指令就允许执行读取操作。而不允许执行该地址块外部的读表指令，它读出的结果为 0。从图 19-4 到 19-6 举例说明了读写表保护。

注：	代码保护位仅能从 “1” 状态改写为 “0” 状态。而不可能从 “0” 状态改写到 “1” 状态。代码保护位只有在整个芯片或块擦写时才能设置为 “1”。而整个芯片或块的擦写又仅可由 ICSP 或外部程序进行。

图 19-4： 不允许写表操作 （**WRTn**）



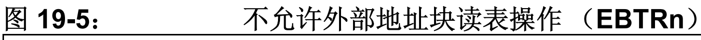
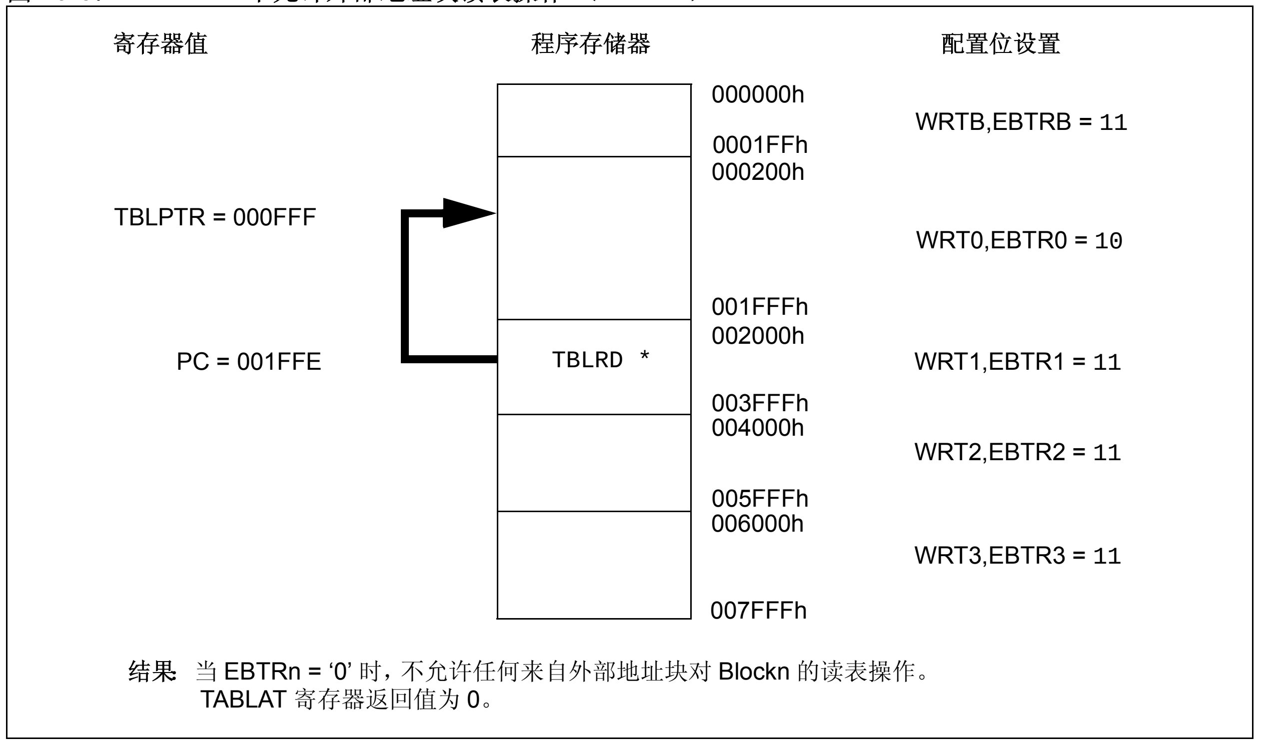


图 19-5： 不允许外部地址块读表操作 （EBTRn）



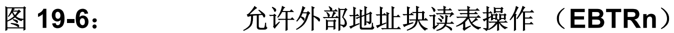
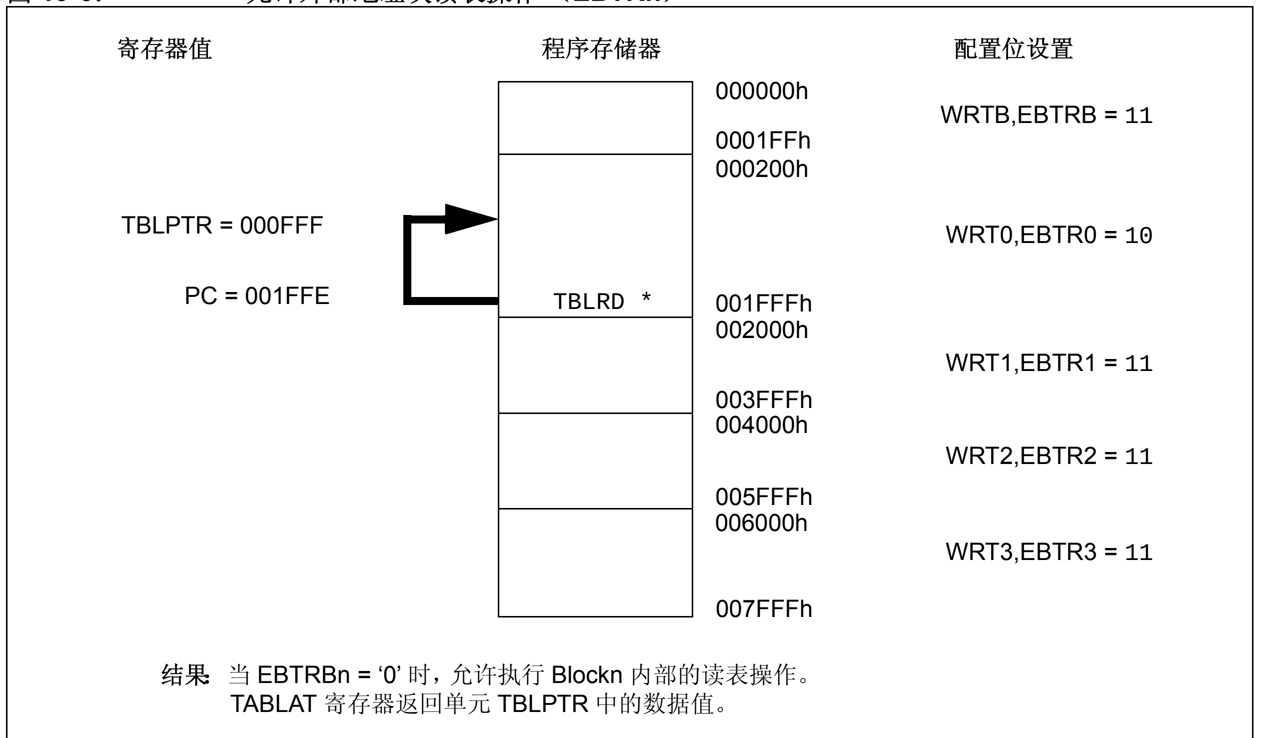


图 19-6： 允许外部地址块读表操作 （EBTRn）



19.4.2 数据 EEPROM 代码保护

整个数据 EEPROM 的外部读写保护通过两个位配置：CPD 和 WRTD。 CPD 禁止了对数据 EEPROM 的外部读写操作。WRTD禁止对数据EEPROM的外部写操作。CPU 可以不受这些保护位的限制，继续读写数据 EEPROM。

19.4.3 配置寄存器保护

配置寄存器具有写保护。 WRTC 位控制对配置寄存器的保护。在用户模式下，WRTC 位只读。WRTC 只能通过 ICSP 或外部程序改写。

19.5 ID 单元

八个存储器地址单元 （200000h-200007h）指定为 ID 地址单元，供用户存储校验和或其他代码标识号。器件正常运行时或在编程 / 校验时，这些单元可以通过 TBLRD 和 TBLWT 指令存取。当器件有代码保护时，可以存取 ID 单元。

对 ID 单元的编程顺序与对闪存存储器的编程顺序相似（参见第 5.5.1 节）。

19.6 在线串行编程

PIC18FXXX 单片机可以在最终应用电路中进行串行编程。只需使用五根线就可以这样做，其中时钟和数据线各 1 根，另外三根线作为电源、接地和编程电压线。这就允许用户采用未编程器件制造电路板，仅在产品交付之前对单片机进行编程。这同样允许对最新的或定制的固件进行编程。

19.7 在线调试器

当配置寄存器 CONFIG4L 中的 DEBUG 位经编程设置为 0 时，使能在线调试功能。这一功能在同 MPLAB® IDE 一起使用的时候，就能进行一些简单的调试。当单片机的这项功能使能时，某些功能部件就不再是通用的。表 19-4显示出哪些功能部件被后台调试器使用了。

表 19-4： 调试器资源

I/O 引脚	RB6, RB7
堆栈	2 层
程序存储器	512 字节
数据存储器	10 字节

要使用单片机的在线调试器功能，设计就必须实现与 MCLR/VPP、VDD、GND、RB7 和 RB6 相连接的在线串行编程。这将与 Microchip 或第三方开发工具公司提供的在线调试器模块交互。

19.8 低电压 ICSP 编程

配置寄存器 CONFIG4L 的 LVP 位对低电压 ICSP 编程使能。这个模式允许通过 ICSP 使用在工作电压范围内的 VDD 电压源对单片机进行编程。这仅意味着 VPP 不必达到 VIHH，但要相应地保持为正常工作电压。在这个模式下， RB5/PGM 引脚专用于编程功能，而不再用作通用的 I/O 引脚。在编程期间，VDD 施加到 MCLR/VPP 引脚。为了进入编程模式，VDD 必须加在 RB5/PGM 引脚上（假定 LVP 位置 1）。出厂时 LVP 位的缺省值为 1。

> 注 **1**：只需将 VIHH 加到 MCLR 引脚，始终可以实现高电压编程模式，这与 LVP 位的状态无关。
>
> **2**：在低电压 ICSP 模式下， RB5 引脚不再用作通用的 I/O 引脚，正常工作期间应保持为低电平以保护电路不受偶发性 ICSP 模式影响。
>
> **3**：当使用低电压 ICSP 编程 （LVP）时，禁止上拉 RB5。如果 TRISB 的 bit 5 被清零，就把 RB5 设置为输出，LATB 的 bit 5 在某些特定的操作下也必须被清零。

如果没有使用低电压编程模式，LVP位可以被编程为0，RB5/PGM引脚作为一个数字I/O引脚。但是，仅在VIHH 接到 MCLR/VPP 引脚上的时候， LVP 位才可被编程改写。

应该注意的是，一旦 LVP 位被编程为 0，只可以在高电压编程模式下工作，也只有通过高电压编程模式对器件编程。

当使用低电压 ICSP 情况下，如果执行批量擦除，该器件的供电电压必须是 4.5V 到 5.5V。这包括代码保护位从开状态重新编程为关状态。对于所有其他低电压 ICSP，该器件可以在正常工作电压下编程。这就意味着可以重新编程 （或增加）唯一的用户 ID 或用户代码。

20.0 指令集综述

PIC18FXXX 指令集与以前的 PICmicro 指令集相比，添加了很多增强功能，同时保持了易于从那些PICmicro指令集移植的特点。

大部分指令只占一个程序存储字的空间（16 位），但有三条指令需要两个程序存储单元。

每条单字指令都是一个 16 位字，由指明指令类型的操作码和进一步说明指令具体操作的一个或多个操作数组成。

整个指令集具有高度正交性，分为四种基本类型：

- **字节**操作类指令
- **位**操作类指令
- **立即数**操作类指令
- **控制**操作类指令

表 20-2中的PIC18FXXX指令集综述列出了**字节**操作类指令、**位**操作类指令、**立即数**操作类指令和**控制**操作类指令。表 20-1 给出了操作码字段的描述。

大部分**字节**操作类指令都有三种操作数：

1. 文件寄存器 （由 “f” 指定）
2. 结果的目标寄存器 （由 “d” 指定）
3. 被访问的存储器 （由 “a” 指定）

文件寄存器标识符 “f” 指定指令会使用哪个文件寄存器。

目标标识符 “d” 指定了操作结果的存放位置。如果 “d” 为 0，操作结果存入 WREG 寄存器中。如果 “d” 为 1，操作结果存入指令指定的文件寄存器中。

所有**位**操作类指令都有三种操作数：

1. 文件寄存器 （由 “f” 指定）
2. 文件寄存器中的位 （由 “b” 指定）
3. 被访问的存储器 （由 “a” 指定）

位段标识符 “b” 选择操作所影响的位，而文件寄存器标识符 “f” 则代表该位所在的文件。

立即数操作类指令可以使用以下部分操作数：

- 装入文件寄存器的立即数值 （由 “k” 指定）
- 希望装入立即数值的 FSR 寄存器 （由 “f” 指定）
- 不需要操作数 （由 “—” 指定）

控制操作类指令可以使用以下部分操作数：

- 程序存储器地址 （由 “n” 指定）
- `Call` 或 `Return` 指令的模式 （由 “s” 指定）
- 读表和写表指令的模式 （由 “m” 指定）
- 不需要操作数 （由 “—” 指定）

除了三条双字指令外，其他指令都是单字指令。这三条指令作为双字指令，因此，所有需要的信息在这 32 位中都可获得。在第二个字中，4 个最高有效位都是 1。如果第二个字作为指令执行 （仅第二个字），它就会作为 `NOP` 指令执行。

除非条件测试为 true 或因执行指令而改变了程序计数器值，否则所有单字指令都是单周期指令。对于前面的情况，执行指令需要两个指令周期，第二个周期中执行的是一条 `NOP` 指令。

执行双字指令需要两个指令周期。

每个指令周期包括 4 个振荡周期。因此，如果振荡器频率为 4 MHz，正常指令执行时间为 1 μs。如果条件测试为 true 或因执行指令而改变了程序计数器值，则该指令的执行时间为 2 μs。双字分支指令（如果为 true）的执行需要 3 μs。

图 20-1 给出了指令的通用格式。

所有示例都使用 “nnh” 格式来表示十六进制数，其中 “h” 代表一个十六进制数。

指令集综述 （见表 20-2）列出了 Microchip 汇编器 （MPASM™）识别的指令。

第 20.1 节描述了每条指令。

表 **20-1**：　操作码字段描述

字段	描述
a	RAM 存取位 a = 0：存取 RAM 内的 RAM 地址 （忽略 BSR 寄存器） a = 1：由 BSR 寄存器指定的 RAM 存储区
bbb	某 8 位文件寄存器内的位地址 （0 到 7）
BSR	存储区选择寄存器。用于选择当前的 RAM 存储区。
d	目标寄存器选择位； d = 0：结果保存至 WREG， d = 1：结果保存至文件寄存器 f。
dest	目标寄存器， WREG 寄存器或指定的寄存器文件地址
f	8 位寄存器文件地址 （0x00 到 0xFF）
fs	12 位寄存器文件地址 （0x000 到 0xFFF）。这是源地址。
fd	12 位寄存器文件地址 （0x000 到 0xFFF）。这是目标地址。
k	立即数、常数或者标号 （可以是 8 位、 12 位或 20 位值）
label	标号名
mm	读表和写表指令的 TBLPTR 寄存器模式。 只与读表和写表指令一起使用：
*	不改变寄存器 （比如带读表和写表的 TBLPTR）
*+	后增寄存器 （比如带读表和写表的 TBLPTR）
*-	后减寄存器 （比如带读表和写表的 TBLPTR）
+*	先增寄存器 （比如带读表和写表的 TBLPTR）
n	相对分支指令的相对地址 （采用二进制补码），或调用 / 分支和返回指令的直接地址
PRODH	乘积的高位字节
PRODL	乘积的低位字节
s	快速调用 / 返回模式选择位。 s = 0： 不对影子寄存器进行更新，也不用影子寄存器的内容更新其他寄存器 s = 1： 其他寄存器和影子寄存器相互更新 （快速模式）
u	未使用或未改变
WREG	工作寄存器 （累加器）
x	与取值无关的位 （0 或 1） 汇编器将产生 x = 0 的代码。为了与所有的 Microchip 软件工具兼容，建议使用这种格式。
TBLPTR	21 位表指针 （指向程序存储器单元）
TABLAT	8 位表锁存器
TOS	栈顶
PC	程序计数器
PCL	程序计数器低字节
PCH	程序计数器高字节
PCLATH	程序计数器高字节锁存器
PCLATU	程序计数器低字节锁存器
GIE	全局中断使能位
WDT	看门狗定时器
$\overline{TO}$	超时标志位
$\overline{PD}$	掉电标志位
C, DC, Z, OV, N	ALU 状态进位标志位、辅助进位标志位、全零标志位、溢出标志位及负标志位
[　]	可选的
(　)	内容
→	赋值给
< >	寄存器位段
∈	表示属于某个集合
italics	用户定义项 （字体为 courier）

图 20-1： 指令的通用格式

针对字节的文件寄存器操作指令 — 指令示例

15 … 10	9	8	7 … 0
OPCODE	d	a	f（文件号）

ADDWF MYREG, W, B

d = 0：结果存入 WREG 寄存器
d = 1：结果存入文件寄存器 (f)
a = 0：强制使用存取存储区
a = 1：根据 BSR 选择存储区
f = 8 位文件寄存器地址

从字节到字节传送操作（双字）

15 … 12	11 … 0
OPCODE	f（源文件号）
1111	f（目标文件号）

MOVFF MYREG1, MYREG2

f = 12 位文件寄存器地址

针对位的文件寄存器操作指令

15 … 12	11 … 9	8	7 … 0
OPCODE	b（位号）	a	f（文件号）

BSF MYREG, bit, B

b = 占 3 位，表示文件寄存器（f）的位位置
a = 0：强制使用存取存储区
a = 1：根据 BSR 选择存储区
f = 8 位文件寄存器地址

立即数操作类指令

15 … 8	7 … 0
OPCODE	k（立即数）

MOVLW 0x7F

k = 8 位立即数值

控制操作类指令

CALL、GOTO 和分支操作指令

15 … 8	7 … 0
OPCODE	n<7:0>（立即数）

15 … 12	11 … 0
1111	n<19:8>（立即数）

GOTO Label

n = 20 位立即数值

15 … 8		7 … 0
OPCODE	S	n<7:0>（立即数）

15 … 12	11 … 0
	n<19:8>（立即数）

CALL MYFUNC

S = 快速位

15 … 11	10 … 0
OPCODE	n<10:0>（立即数）

BRA MYFUNC

15 … 8	7 … 0
OPCODE	n<7:0>（立即数）

BC MYFUNC

表 **20-2**： **PIC18FXXX** 指令集

助记符、操作数		描述	周期	16 位宽指令字				受影响的状态位	备注
				MSb			LSb		
针对字节的文件寄存器操作									
ADDWF	f, d, a	WREG 与 f 相加	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	WREG 与 f 及进位相加	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	WREG 和 f 进行 “与”操作	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	寄存器 f 清零	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	对 f 取反	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	将 f 与 WREG 比较， = 则跳过	1 (2 或 3)	0110	001a	ffff	ffff	无	4
CPFSGT	f, a	将 f 与 WREG 比较， > 则跳过	1 (2 或 3)	0110	010a	ffff	ffff	无	4
CPFSLT	f, a	将 f 与 WREG 比较， < 则跳过	1 (2 或 3)	0110	000a	ffff	ffff	无	1, 2
DECF	f, d, a	f 减 1	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	f 减 1，为 0 则跳过	1 (2 或 3)	0010	11da	ffff	ffff	无	1, 2, 3, 4
DCFSNZ	f, d, a	f 减 1，非 0 则跳过	1 (2 或 3)	0100	11da	ffff	ffff	无	1, 2
INCF	f, d, a	f 加 1	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	f 加 1，为 0 则跳过	1 (2 或 3)	0011	11da	ffff	ffff	无	4
INFSNZ	f, d, a	f 加 1，非 0 则跳过	1 (2 或 3)	0100	10da	ffff	ffff	无	1, 2
IORWF	f, d, a	WREG 和 f 进行 “或”操作	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	传送寄存器 f 的内容	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	将 f_s （源）传送到 （第一个字）	2	1100	ffff	ffff	ffff	无	
		f_d （目标）（第二个字）		1111	ffff	ffff	ffff		
MOVWF	f, a	WREG 的内容传送到 f	1	0110	111a	ffff	ffff	无	
MULWF	f, a	WREG 和 f 相乘	1	0000	001a	ffff	ffff	无	
NEGF	f, a	对 f 求补	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	1, 2
RLCF	f, d, a	带进位循环左移 f	1	0011	01da	ffff	ffff	C, Z, N	
RLNCF	f, d, a	循环左移 f （无进位）	1	0100	01da	ffff	ffff	Z, N	1, 2
RRCF	f, d, a	带进位循环右移 f	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	循环右移 f （无进位）	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	置位 f	1	0110	100a	ffff	ffff	无	
SUBFWB	f, d, a	WREG 减去 f 和借位	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWF	f, d, a	f 减去 WREG	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	
SUBWFB	f, d, a	f 减去 WREG 和借位	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	1, 2
SWAPF	f, d, a	f 半字节交换	1	0011	10da	ffff	ffff	无	4
TSTFSZ	f, a	测试 f，为 0 时跳过	1 (2 或 3)	0110	011a	ffff	ffff	无	1, 2
XORWF	f, d, a	WREG 和 f 进行 “异或”操作	1	0001	10da	ffff	ffff	Z, N	
针对位的文件寄存器操作指令									
BCF	f, b, a	f 的 bit b 清零	1	1001	bbba	ffff	ffff	无	1, 2
BSF	f, b, a	f 的 bit b 置 1	1	1000	bbba	ffff	ffff	无	1, 2
BTFSC	f, b, a	检测 f 的 bit b，为 0 则跳过	1 (2 或 3)	1011	bbba	ffff	ffff	无	3, 4
BTFSS	f, b, a	检测 f 的 bit b，为 1 则跳过	1 (2 或 3)	1010	bbba	ffff	ffff	无	3, 4
BTG	f, d, a	f 的 bit b 取反	1	0111	bbba	ffff	ffff	无	1, 2

注 **1**：当端口寄存器用自身内容修改自身时 （例如， `MOVF PORTB, 1, 0`），使用的值是引脚上的当前值。例如，如果一引脚配置为输入，其数据锁存器中的值为 “1”，但此时外部器件将该引脚拉为低电平，则将数据 0 写回数据锁存器。

2：当对 TMR0 寄存器执行该指令 （以及 d=1）时，如果预分频器被指定为 TMR0，则将其清零。

3：如果程序计数器 （PC）被修改或者条件检测为 true，则该指令需要两个指令周期。第二个指令周期执行一条 `NOP` 指令。

4：某些指令是双字指令。如果指令的第一个字无法取得第二个字 16 位中的信息，则第二个字将作为 `NOP` 指令执行。这将确保所有程序存储器地址上存储的都是合法的指令。

5：如果写表操作开始对内部存储器进行写入，则写操作将持续到终止为止。

表 20-2： PIC18FXXX 指令集 （续）

助记符、操作数		描述	周期	16 位宽指令字				受影响的状态位	备注
				MSb			LSb		
控制操作									
BC	n	进位则转移	1 (2)	1110	0010	nnnn	nnnn	无	
BN	n	为负则转移	1 (2)	1110	0110	nnnn	nnnn	无	
BNC	n	无进位则转移	1 (2)	1110	0011	nnnn	nnnn	无	
BNN	n	不为负则转移	1 (2)	1110	0111	nnnn	nnnn	无	
BNOV	n	不溢出则转移	1 (2)	1110	0101	nnnn	nnnn	无	
BNZ	n	不为零则转移	2	1110	0001	nnnn	nnnn	无	
BOV	n	溢出则转移	1 (2)	1110	0100	nnnn	nnnn	无	
BRA	n	无条件转移	1 (2)	1101	0nnn	nnnn	nnnn	无	
BZ	n	为零则转移	1 (2)	1110	0000	nnnn	nnnn	无	
CALL	n, s	调用子程序的第一个字	2	1110	110s	kkkk	kkkk	无	
		第二个字		1111	kkkk	kkkk	kkkk		
CLRWDT	—	看门狗定时器清零	1	0000	0000	0000	0100	$\overline{TO}$，$\overline{PD}$	
DAW	—	十进制调整 WREG	1	0000	0000	0000	0111	C	
GOTO	n	跳转到地址第一个字	2	1110	1111	kkkk	kkkk	无	
		第二个字		1111	kkkk	kkkk	kkkk		
NOP	—	无操作	1	0000	0000	0000	0000	无	
NOP	—	无操作	1	1111	xxxx	xxxx	xxxx	无	4
POP	—	将返回堆栈顶部的内容弹出 (TOS)	1	0000	0000	0000	0110	无	
PUSH	—	将内容压入返回堆栈的顶部 (TOS)	1	0000	0000	0000	0101	无	
RCALL	n	相对调用	2	1101	1nnn	nnnn	nnnn	无	
RESET		软件器件复位	1	0000	0000	1111	1111	全部	
RETFIE	s	中断返回	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	返回时将立即数送入 WREG	2	0000	1100	kkkk	kkkk	无	
RETURN	s	从子程序返回	2	0000	0000	0001	001s	无	
SLEEP	—	进入休眠模式	1	0000	0000	0000	0011	$\overline{TO}$，$\overline{PD}$	

注 1： 当端口寄存器用自身内容修改自身时 （例如， `MOVF PORTB, 1, 0`），使用的值是引脚上的当前值。例如，如果一引脚配置为输入，其数据锁存器中的值为 “1”，但此时外部器件将该引脚拉为低电平，则将数据 0 写回数据锁存器。

2： 当对 TMR0 寄存器执行该指令 （以及 d=1）时，如果预分频器被指定为 TMR0，则将其清零。

3： 如果程序计数器 （PC）被修改或者条件检测为 true，则该指令需要两个指令周期。第二个指令周期执行一条 `NOP` 指令。

4： 某些指令是双字指令。如果指令的第一个字无法取得第二个字 16 位中的信息，则第二个字将作为 `NOP` 指令执行。这将确保所有程序存储器地址上存储的都是合法的指令。

5： 如果写表操作开始对内部存储器进行写入，则写操作将持续到终止为止。

表 **20-2**： **PIC18FXXX** 指令集 （续）

助记符、 操作数		描述	周期	16 位宽指令字 MSb			LSb	受影响的 状态位	备注
立即数操作									
ADDLW	k	WREG 与立即数相加	1	0000	1111	kkkk	kkkk	C, DC, Z, OV,N	
ANDLW	k	WREG 和立即数进行 “与”操作	1	0000	1011	kkkk	kkkk	Z, N	
IORLW	k	WREG 和立即数进行 “或”操作	1	0000	1001	kkkk	kkkk	Z, N	
LFSR	f, k	立即数 （12 位）第二个字发送 到 FSRx 第一个字	2	1110 1111	1110 0000	00ff kkkk	kkkk kkkk	无	
MOVLB	k	立即数发送到 BSR<3:0>	1	0000	0001	0000	kkkk	无	
MOVLW	k	立即数发送到 WREG	1	0000	1110	kkkk	kkkk	无	
MULLW	k	WREG 和立即数相乘	1	0000	1101	kkkk	kkkk	无	
RETLW	k	返回时将立即数送入 WREG	2	0000	1100	kkkk	kkkk	无	
SUBLW	k	立即数减去 WREG	1	0000	1000	kkkk	kkkk	C, DC, Z, OV,N	
XORLW	k	WREG 和立即数进行 “异或”操作	1	0000	1010	kkkk	kkkk	Z, N	
数据存储器 ↔ 程序存储器操作									
TBLRD*		读表	2	0000	0000	0000	1000	无	
TBLRD*+		读表，然后增 1		0000	0000	0000	1001	无	
TBLRD*-		读表，然后减 1		0000	0000	0000	1010	无	
TBLRD+*		增 1，然后读表		0000	0000	0000	1011	无	
TBLWT*		写表	2 (5)	0000	0000	0000	1100	无	
TBLWT*+		写表，然后增 1		0000	0000	0000	1101	无	
TBLWT*-		写表，然后减 1		0000	0000	0000	1110	无	
TBLWT+*		增 1，然后写表		0000	0000	0000	1111	无	

注 **1**：当端口寄存器用自身内容修改自身时 （例如，`MOVF PORTB, 1, 0`），使用的值是引脚上的当前值。例如，如果一引脚配置为输入，其数据锁存器中的值为 “1”，但此时外部器件将该引脚拉为低电平，则将数据 0 写回数据锁存器。

2：当对 TMR0 寄存器执行该指令 （以及 d=1）时，如果预分频器被指定为 TMR0，则将其清零。

3：如果程序计数器 （PC）被修改或者条件检测为 true，则该指令需要两个指令周期。第二个指令周期执行一条 `NOP` 指令。

4：某些指令是双字指令。如果指令的第一个字无法取得第二个字 16 位中的信息，则第二个字将作为 `NOP` 指令执行。这将确保所有程序存储器地址上存储的都是合法的指令。

5：如果写表操作开始对内部存储器进行写入，则写操作将持续到终止为止。

20.1 指令集

ADDLW 立即数与 W 相加

语法： [*label*] ADDLW k

操作数： 0 ≤ k≤255

操作： (W) + k → W

受影响的状态位： N, OV, C, DC, Z

机器码：

0000	1111	kkkk	kkkk

描述： 将 W 的内容与 8 位立即数 k 相加，结果存入 W。

指令字： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 k	处理 数据	写入 W

示例： `ADDLW 0x15`

执行指令前
W = 0x10

执行指令后
W = 0x25

ADDWF W 与 f 相加

语法： [*label*] ADDWF f [,d [,a]]

操作数： 0≤ f ≤255
d ∈ [0,1]
a ∈[0,1]

操作： (W) + (f) → dest

受影响的状态位： N, OV, C, DC, Z

机器码：

0010	01da	ffff	ffff

描述： 将 W 与寄存器 f 相加。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回寄存器 f（默认情况）。如果 a 为 0，选择存取存储区。如果 a 为 1，则使用 BSR 寄存器。

指令字： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理 数据	写入 目标寄存器

示例： `ADDWF REG, 0, 0`

执行指令前
W = 0x17
REG = 0xC2

执行指令后
W = 0xD9
REG = 0xC2

ADDWFC　W 和 f 及进位相加

语法：	[*label*] ADDWFC　f [,d [,a]]
操作数：	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
操作：	(W) + (f) + (C) → dest
受影响的状态位：	N,OV, C, DC, Z
机器码：	0010 | 00da | ffff | ffff
描述：	将 W、进位标志位和数据存储单元 f 相加。如果 d 为 0，结果存入 W。如果 d 为 1，结果存入数据存储单元 f。如果 a 为 0，则选择存取存储区。如果 a 为 1，则使用 BSR 寄存器。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理 数据	写入 目标寄存器

示例：　`ADDWFC　REG, 0, 1`

执行指令前
- 进位 = 1
- REG = 0x02
- W = 0x4D

执行指令后
- 进位 = 0
- REG = 0x02
- W = 0x50

ANDLW　立即数和 W 进行 “与” 操作

语法：	[*label*]　ANDLW　k
操作数：	$0 \leq k \leq 255$
操作：	(W) .AND. k → W
受影响的状态位：	N,Z
机器码：	0000 | 1011 | kkkk | kkkk
描述：	将 W 的内容和 8 位立即数 k 进行 “与” 操作。结果保入 W。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理 数据	写入 W

示例：　`ANDLW　0x5F`

执行指令前
- W = 0xA3

执行指令后
- W = 0x03

ANDWF **W 和 f 进行 “与”操作**

语法：	[*label*] ANDWF f [,d [,a]]
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(W) .AND. (f) → dest
受影响的状态位：	N,Z
机器码：	`0001` `01da` `ffff` `ffff`
描述：	将 W 的内容和寄存器 f 进行 “与”操作。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f （默认情况）。如果 a 为 0，选择存取存储区。如果 a 为 1，则使用 BSR （默认情况）。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理 数据	写入 目标寄存器

示例： `ANDWF REG, 0, 0`

执行指令前
W = 0x17
REG = 0xC2

执行指令后
W = 0x02
REG = 0xC2

BC **进位则转移**

语法：	[*label*] BC n
操作数：	-128≤ n ≤127
操作：	如果进位为 1 (PC) + 2 + 2n → PC
受影响的状态位：	无
机器码：	`1110` `0010` `nnnn` `nnnn`
描述：	如果进位为 1，程序转移。 二进制补码 “2n”与 PC 相加，结果存入 PC。因为 PC 要加 1 才能取下一条指令，因此新地址是 PC+2+2n。因而，该指令是一条双周期指令。
指令字：	1
指令周期：	1(2)

Q 周期操作：
如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理 数据	写入 PC
无 操作	无 操作	无 操作	无 操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理 数据	无 操作

示例： `HERE BC 5`

执行指令前
PC = 地址 (HERE)

执行指令后
如果进位 = 1;
PC = 地址 (HERE+12)
如果进位 = 0;
PC = 地址 (HERE+2)

BCF f 的 bit b 清零

语法： [*label*] BCF f,b[,a]

操作数： $0 \leq f \leq 255$
$0 \leq b \leq 7$
$a \in [0,1]$

操作： 0→f<b>

受影响的状态位： 无

机器码：

1001	bbba	ffff	ffff

描述： 将寄存器 f 的 bit b 清零。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则会根据 BSR 的值选择存储区 （默认情况）。

指令字： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理 数据	写 寄存器 f

示例： `BCF FLAG_REG, 7, 0`

执行指令前
FLAG_REG = 0xC7

执行指令后
FLAG_REG = 0x47

BN 为负则转移

语法： [*label*] BN n

操作数： $-128 \leq n \leq 127$

操作： 如果负数位为 1
(PC) + 2 + 2n→ PC

受影响的状态位： 无

机器码：

1110	0110	nnnn	nnnn

描述： 如果负数位为 1，程序转移。二进制补码 “2n”与 PC 相加，结果存入 PC。因为 PC 需要加 1 才能取下一条指令，因此新地址将是 PC+2+2n。该指令是一条双周期指令。

指令字： 1

指令周期： 1(2)

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理 数据	写入 PC
无 操作	无 操作	无 操作	无 操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理 数据	无 操作

示例： `HERE BN Jump`

执行指令前
PC = 地址 (HERE)

执行指令后
如果负数位 = 1;
PC = 地址 (Jump)
如果负数位 = 0;
PC = 地址 (HERE+2)

BNC 无进位则转移

语法：	[*label*] BNC n
操作数：	-128 ≤ n ≤ 127
操作：	如果进位为 0 (PC) + 2 + 2n → PC
受影响的状态位：	无
机器码：	1110 0011 nnnn nnnn
描述：	如果进位为 0，程序转移。 二进制补码 “2n”与 PC 相加，结果存入 PC。由于 PC 需要加 1 才能取下一条指令，所以新地址是 PC+2+2n。这一指令是一条双周期指令。
指令字：	1
指令周期：	1(2)

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
无操作	无操作	无操作	无操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	无操作

示例： `HERE BNC Jump`

执行指令前

PC = 地址 (HERE)

执行指令后

如果进位 = 0;
PC = 地址 (Jump)
如果进位 = 1;
PC = 地址 (HERE+2)

BNN 不为负则转移

语法：	[*label*] BNN n
操作数：	-128 ≤ n ≤127
操作：	如果负数位为 0 (PC) + 2 + 2n →PC
受影响的状态位：	无
机器码：	1110 0111 nnnn nnnn
描述：	如果负数位为 1，程序将转移。 二进制补码 “2n”与 PC 相加，结果存入 PC。由于 PC 需要加 1 才能取下一条指令，所以新地址是 PC+2+2n。这一指令是一条双周期指令。
指令字：	1
指令周期：	1(2)

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
无操作	无操作	无操作	无操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	无操作

示例： `HERE BNN Jump`

执行指令前

PC = 地址 (HERE)

执行指令后

如果负数位 = 0;
PC = 地址 (Jump)
如果负数位 = 1;
PC = 地址 (HERE+2)

BNOV 不溢出则转移

语法：	[*label*] BNOV n
操作数：	-128 ≤ n ≤127
操作：	如果溢出位为 0 (PC) + 2 + 2n → PC
受影响的状态位：	无
机器码：	1110 0101 nnnn nnnn
描述：	如果溢出位为 0，程序将转移。二进制补码 “2n”与 PC 相加，结果存入 PC。因为 PC 需要加 1 才能取下一条指令，所以新地址是 PC+2+2n。这一指令是一条双周期指令。
指令字：	1
指令周期：	1(2)

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
无操作	无操作	无操作	无操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	无操作

示例： `HERE BNOV Jump`

执行指令前
- PC = 地址 (HERE)

执行指令后
- 如果溢出位 = 0;
- PC = 地址 (Jump)
- 如果溢出位 = 1;
- PC = 地址 (HERE+2)

BNZ 不为零则转移

语法：	[*label*] BNZ n
操作数：	-128 ≤ n ≤ 127
操作：	如果全零位为 0 (PC) + 2 + 2n→PC
受影响的状态位：	无
机器码：	1110 0001 nnnn nnnn
描述：	如果全零位为 0，程序将转移。二进制补码 “2n”与 PC 相加，结果存入 PC。因为 PC 需要加 1 才能取下一条指令，所以新地址是 PC+2+2n。这一指令是一条双周期指令。
指令字：	1
指令周期：	1(2)

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
无操作	无操作	无操作	无操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	无操作

示例： `HERE BNZ Jump`

执行指令前
- PC = 地址 (HERE)

执行指令后
- 如果全零位 = 0;
- PC = 地址 (Jump)
- 如果全零位 = 1;
- PC = 地址 (HERE+2)

BRA 无条件转移

语法： [*label*] BRA n

操作数： $-1024 \le n \le 1023$

操作： $(PC) + 2 + 2n \rightarrow PC$

受影响的状态位： 无

机器码：

1101	0nnn	nnnn	nnnn

描述： 将二进制补码 “2n”与 PC 相加，结果存入 PC。因为 PC 需要加 1 才能取下一条指令，所以新地址是 PC+2+2n。该指令是一条双周期指令。

指令字： 1

指令周期： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
无操作	无操作	无操作	无操作

示例： `HERE BRA Jump`

执行指令前
PC = 地址 (HERE)

执行指令后
PC = 地址 (Jump)

BSF f 的 bit b 置 1

语法： [*label*] BSF f,b[,a]

操作数： $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

操作： $1 \rightarrow f<b>$

受影响的状态位： 无

机器码：

1000	bbba	ffff	ffff

描述： 将寄存器 f 的 bit b 置 1。如果 a 为 0，则选择存取存储区，并忽略 BSR 的值。如果 a 为 1，则根据 BSR 的值选择存取存储区。

指令字： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 f	处理数据	写入寄存器 f

示例： `BSF FLAG_REG, 7, 1`

执行指令前
FLAG_REG = 0x0A

执行指令后
FLAG_REG = 0x8A

BTFSC 检测 f 的 bit b，为 0 则跳过

语法：	[*label*] BTFSC f,b[,a]
操作数：	0 ≤ f ≤ 255 0 ≤b ≤ 7 a ∈ [0,1]
操作：	如果 (f<b>) = 0，跳过
受影响的状态位：	无
机器码：	1011 bbba ffff ffff
描述：	如果寄存器 f 的 bit b 为 0，则跳过下一条指令。 如果 bit b 为 0，那么 （在当前指令执行期间所取的）下一条指令不再执行，改为执行一条 NOP 指令，使该指令变成双周期指令。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则会根据 BSR 的值选择存储区 （默认情况）。
指令字：	1
指令周期：	1(2) 注： 如果跳过一条指令，继续执行一条双字指令，则为 3 周期指令。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	无操作

如果跳过：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作

如果跳过一条指令后继续执行双字指令：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作
无操作	无操作	无操作	无操作

示例：

```
HERE    BTFSC   FLAG, 1, 0
FALSE   :
TRUE    :
```

执行指令前
PC = 地址 (HERE)

执行指令后
如果标志位 <1>= 0;
PC = 地址 (TRUE)
如果标志位 <1>= 1;
PC = 地址 (FALSE)

BTFSS 检测 f 的 bit b，为 1 则跳过

语法：	[*标号*] BTFSS f,b[,a]
操作数：	0 ≤ f ≤ 255 0≤ b ≤ 7 a ∈ [0,1]
操作：	如果 (f<b>) = 1，跳过
受影响的状态位：	无
机器码：	1010 bbba ffff ffff
描述：	如果寄存器 f 的 bit b 为 1，则跳过下一条指令。 如果 bit b 为 1，那么 （在当前指令执行期间所取的）下一条指令不再执行，改为执行一条 NOP 指令，使该指令变成双周期指令。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则根据 BSR 的值选择存储区 （默认情况）。
指令字：	1
指令周期：	1(2) 注： 如果跳过一条指令，继续执行一条双字指令，则为 3 周期指令。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	无操作

如果跳过：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作

如果跳过一条指令后继续执行双字指令：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作
无操作	无操作	无操作	无操作

示例：

```
HERE    BTFSS   FLAG, 1, 0
FALSE   :
TRUE    :
```

执行指令前
PC = 地址 (HERE)

执行指令后
如果标志位 <1>= 0;
PC = 地址 (FALSE)
如果标志位 <1>= 1;
PC = 地址 (TRUE)

BTG　f 的 bit b 取反

语法：	[*label*] BTG f,b[,a]
操作数：	$0 \le f \le 255$ $0 \le b \le 7$ $a \in [0,1]$
操作：	$(\overline{f<b>}) \rightarrow f<b>$
受影响的状态位：	无
机器码：	0111 bbba ffff 0111
描述：	对数据存储地址单元 f 的 bit b 取反。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则根据 BSR 的值选择存储区 （默认情况）。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理 数据	写 寄存器 f

示例： `BTG PORTC, 4, 0`

执行指令前：
PORTC = 0111 0101 [0x75]

执行后：
PORTC = 0110 0101 [0x65]

BOV　溢出则转移

语法：	[*label*] BOV n
操作数：	$-128 \le n \le 127$
操作：	如果溢出位为 1 $(PC) + 2 + 2n \rightarrow PC$
受影响的状态位：	无
机器码：	1110 0100 nnnn nnnn
描述：	如果溢出位为 1，程序转移。二进制补码 “2n”与 PC 相加，结果存入 PC。由于 PC 要加 1 才能取到下一条指令，所以新地址是 PC+2+2n。这一指令是一条双周期指令。
指令字：	1
指令周期：	1(2)

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理 数据	写入 PC
无 操作	无 操作	无 操作	无 操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理 数据	无 操作

示例： `HERE BOV Jump`

执行指令前
PC = 地址 (HERE)

执行指令后
如果溢出位 = 1;
PC = 地址 (Jump)
如果溢出位 = 0;
PC = 地址 (HERE+2)

BZ 为零则转移

语法：	[*label*] BZ n
操作数：	-128 ≤ n ≤ 127
操作：	如果全零位为 1 (PC) + 2 + 2n → PC
受影响的状态位：	无
机器码：	1110 0000 nnnn nnnn
描述：	如果全零位为 1，程序转移。 二进制补码 “2n” 与 PC 相加，结果存入 PC。由于 PC 要加 1 才能取下一条指令，所以新地址是 PC+2+2n。这一指令是一条双周期指令。
指令字：	1
指令周期：	1(2)

Q 周期操作：

如果跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	写入 PC
无操作	无操作	无操作	无操作

如果不跳转：

Q1	Q2	Q3	Q4
译码	读立即数 n	处理数据	无操作

示例：

```
HERE        BZ   Jump
```

执行指令前

PC = 地址 (HERE)

执行指令后

如果全零位 = 1;

PC = 地址 (Jump)

如果全零位 = 0;

PC = 地址 (HERE+2)

CALL 子程序调用

语法：	[*label*] CALL k [,s]
操作数：	0 ≤ k ≤ 1048575 s ∈ [0,1]
操作：	(PC) +4→ TOS, k →PC<20:1>, 如果 s = 1 (W) →WS, (STATUS)→ STATUSS, (BSR) → BSRS
受影响的状态位：	无
机器码： 第 1 个字（k<7:0>）	1110 110s $k_7$kkk kkkk$_0$
第 2 个字（k<19:8>）	1111 k_{19}kkk kkkk kkkk$_8$
描述：	2 MB 存储空间内的子程序调用。首先，将返回地址 （PC+ 4）压入返回堆栈。如果 s 为 1，则 W、STATUS 和 BSR 寄存器也会被压入对应的影子寄存器 WS、 STATUSS 和 BSRS。如果 s 为 0，不会产生更新 （默认情况）。然后，将 20 位值 k 装入 PC<20:1>。 CALL 为双周期指令。
指令字：	2
指令周期：	2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k<7:0>，	将 PC 压入堆栈	读立即数 k<19:8>，写入 PC
无操作	无操作	无操作	无操作

示例：

```
HERE      CALL   THERE,1
```

执行指令前

PC = 地址 (HERE)

执行指令后

PC = 地址 (THERE)
TOS = 地址 (HERE + 4)
WS = W
BSRS = BSR
STATUSS= STATUS

CLRF **寄存器 f 清零**

语法：	[*label*] CLRF f [,a]
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]
操作：	000h → f 1 → Z
受影响的状态位：	Z
机器码：	`0110` `101a` `ffff` `ffff`
描述：	指定寄存器的内容清零。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则会根据 BSR 的值选择存储区（默认情况）。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理 数据	写入 寄存器 f

示例： `CLRF FLAG_REG,1`

执行指令前
FLAG_REG = 0x5A

执行指令后
FLAG_REG = 0x00

CLRWDT **看门狗定时器清零**

语法：	[*label*] CLRWDT
操作数：	无
操作：	000h → WDT, 000h → WDT 后分频器 , 1→$\overline{TO}$, 1 → $\overline{PD}$
受影响的状态位：	$\overline{TO}$，$\overline{PD}$
机器码：	`0000` `0000` `0000` `0100`
描述：	CLRWDT 指令复位看门狗定时器。而且会复位 WDT 的后分频器。状态位 $\overline{TO}$ 和 $\overline{PD}$ 被置 1。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	无 操作	处理 数据	无 操作

示例： `CLRWDT`

执行指令前
WDT 计数器 = ?

执行指令后
WDT 计数器 = 0x00
WDT 后分频器 = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF 对 f 取反

语法：	[*label*] COMF f [,d [,a]]
操作数：	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
操作：	$(\overline{f}) \rightarrow dest$
受影响的状态位：	N, Z
机器码：	`0001` `11da` `ffff` `ffff`
描述：	对寄存器 f 的内容取反。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f （默认情况）。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则根据 BSR 的值选择存储区 （默认情况）。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理 数据	写入 目标寄存器

示例：

```
COMF    REG, 0, 0
```

执行指令前
REG = 0x13

执行指令后
REG = 0x13
W = 0xEC

CPFSEQ 将 f 与 W 比较，如果 f = W 则跳过

语法：	[*label*] CPFSEQ f [,a]
操作数：	$0 \le f \le 255$ $a \in [0,1]$
操作：	(f) − (W), 如果 (f) = (W) 则跳过 （无符号比较）
受影响的状态位：	无
机器码：	`0110` `001a` `ffff` `ffff`
描述：	执行无符号减法，比较数据存储单元 f 和 W 中的内容。 如果 'f' = W， 那么不再执行取得的指令，转而执行 NOP 指令，从而使该指令变成双周期指令。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则根据 BSR 的值选择存储区 （默认情况）。
指令字：	1
指令周期：	1(2) 注：如果跳过一条指令，然后继续执行双字指令，则为 3 周期指令。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理数据	无 操作

如果跳过：

Q1	Q2	Q3	Q4
无 操作	无 操作	无 操作	无 操作

如果跳过，之后执行双字指令：

Q1	Q2	Q3	Q4
无 操作	无 操作	无 操作	无 操作
无 操作	无 操作	无 操作	无 操作

示例：

```
HERE      CPFSEQ REG, 0
NEQUAL    :
EQUAL     :
```

执行指令前
PC 地址 = HERE
W = ?
REG = ?

执行指令后
如果 REG = W；
PC = 地址 (EQUAL)
如果 REG ≠ W;
PC = 地址 (NEQUAL)

CPFSGT　　将 f 与 W 比较，如果 f>W 则跳过

语法：　[*label*] CPFSGT f [,a]

操作数：　0≤f ≤255
a ∈ [0,1]

操作：　(f) – (W),
如果 (f) > (W) 则跳过
（无符号比较）

受影响的状态位：　无

机器码：

0110	010a	ffff	ffff

描述：　执行无符号减法，对数据存储单元 f 和 W 中的内容进行比较。
如果 f 的内容大于 WREG 的内容，那么不再执行取到的指令，转而执行一条 NOP 指令，从而使该指令变成双周期指令。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则根据 BSR 的值选择存储区 （默认情况）。

指令字：　1

指令周期：　1(2)

注：　如果跳过一条指令，继续执行一条双字指令，则为 3 周期指令。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理 数据	无 操作

如果跳过：

Q1	Q2	Q3	Q4
无 操作	无 操作	无 操作	无 操作

如果跳过后执行双字指令：

Q1	Q2	Q3	Q4
无 操作	无 操作	无 操作	无 操作
无 操作	无 操作	无 操作	无 操作

示例：

```
HERE      CPFSGT REG, 0
NGREATER  :
GREATER   :
```

执行指令前
PC = 地址 (HERE)
W = ?

执行指令后
如果 REG > W;
PC = 地址 (GREATER)
如果 REG ≤ W;
PC = 地址 (NGREATER)

CPFSLT　　将 f 与 W 比较，如果 f<W 则跳过

语法：　[*label*] CPFSLT f [,a]

操作数：　0≤ f ≤ 255
a ∈ [0,1]

操作：　(f) - (W),
如果 (f) < (W) 则跳过
（无符号比较）

受影响的状态位：　无

机器码：

0110	000a	ffff	ffff

描述：　执行无符号的减法，对数据存储单元 f 和 W 中的内容进行比较。
如果 f 的内容小于 W 的内容，那么不再执行取到的指令，转而执行一条 NOP 指令，从而使该指令变成双周期指令。如果 a 为 0，则选择存取存储区。如果 a 为 1，则使用 BSR （默认情况）。

指令字：　1

指令周期：　1(2)

注：　如果跳过一条指令，继续执行一条双字指令，则为 3 周期指令。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理 数据	无 操作

如果跳过：

Q1	Q2	Q3	Q4
无 操作	无 操作	无 操作	无 操作

如果跳过后执行双字指令：

Q1	Q2	Q3	Q4
无 操作	无 操作	无 操作	无 操作
无 操作	无 操作	无 操作	无 操作

示例：

```
HERE    CPFSLT REG, 1
NLESS   :
LESS    :
```

执行指令前
PC = 地址 (HERE)
W = ?

执行指令后
如果 REG < W;
PC = 地址 (LESS)
如果 REG ≥ W;
PC = 地址 (NLESS)

DAW　十进制调整 W 寄存器

语法：　[*label*]　DAW

操作数：　无

操作：　如果 [W<3:0> >9] 或 [DC = 1] 那么
(<3:0>) + 6 → W<3:0>;
否则
(W<3:0>) → W<3:0>;

如果 [W<7:4> >9] 或 [C = 1] 那么
(W<7:4>) + 6 → W<7:4>;
否则
(W<7:4>) → W<7:4>;

受影响的状态位：　C

机器码：

0000	0000	0000	0111

描述：　DAW 调整 W 内的 8 位值，这 8 位值为前面两个变量 （格式均为紧凑型 BCD 格式）的和，并产生正确的紧凑型 BCD 格式的结果。

指令字：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 W	处理 数据	写入 W

示例 1：　DAW

执行指令前
W = 0xA5
C = 0
DC = 0

执行指令后
W = 0x05
C = 1
DC = 0

示例 2：

执行指令前
W = 0xCE
C = 0
DC = 0

执行指令后
W = 0x34
C = 1
DC = 0

DECF　f 减 1

语法：　[*label*]　DECF　f [,d [,a]]

操作数：　$0 \leq f \leq 255$
$d \in [0,1]$
$a \in [0,1]$

操作：　(f) − 1 → dest

受影响的状态位：　C, DC, N, OV, Z

机器码：

0000	01da	ffff	ffff

描述：　寄存器 f 内容减 1。如果 d 为 0，结果存入 W 寄存器。如果 d 为 1，结果存回寄存器 f （默认情况）。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则会根据 BSR 的值选择存储区 （默认情况）。

指令字：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 f	处理 数据	写入 目标寄存器

示例：　DECF　CNT,　1, 0

执行指令前
CNT = 0x01
Z = 0

执行指令后
CNT = 0x00
Z = 1

DECFSZ f 减 1，为 0 则跳过

语法：	[*label*] DECFSZ f [,d [,a]]
操作数：	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
操作：	(f) - 1 → dest， 结果为 0 则跳过
受影响的状态位：	无
机器码：	`0010` `11da` `ffff` `ffff`
描述：	寄存器 f 的内容减 1。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回寄存器 “f”（默认情况）。 如果结果为 0，则不再执行取得的下一条指令，转而执行一条 NOP 指令，从而该指令变成双周期指令。 如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 的值选择存储区 （默认情况）。
指令字：	1
指令周期：	1(2) **注：** 如果跳过一条指令，继续执行一条双字指令，则为 3 周期指令。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	写入目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作

如果跳过且后面跟有双字指令：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作
无操作	无操作	无操作	无操作

示例：

```
HERE        DECFSZ   CNT, 1, 1
            GOTO     LOOP
CONTINUE
```

执行指令前
PC = 地址 (HERE)

执行指令后
CNT = CNT - 1
如果 CNT = 0;
PC = 地址 (CONTINUE)
如果 CNT ≠ 0;
PC = 地址 (HERE+2)

DCFSNZ f 减 1，非 0 则跳过

语法：	[*label*] DCFSNZ f [,d [,a]]
操作数：	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
操作：	(f) - 1 → dest， 结果非 0 则跳过
受影响的状态位：	无
机器码：	`0100` `11da` `ffff` `ffff`
描述：	寄存器 f 的内容减 1。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回寄存器 f （默认情况）。 如果结果非 0，则不再执行取得的下一条指令，转而执行一条 NOP 指令，从而该指令变成双周期指令。 如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 的值选择存储区 （默认情况）。
指令字：	1
指令周期：	1(2) **注：** 如果跳过一条指令，继续执行一条双字指令，则为 3 周期指令。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	写入目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作

如果跳过且后面跟有双字指令：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作
无操作	无操作	无操作	无操作

示例：

```
HERE    DCFSNZ  TEMP, 1, 0
ZERO    :
NZERO   :
```

执行指令前
TEMP = ?

执行指令后
TEMP = TEMP - 1，
如果 TEMP = 0;
PC = 地址 (ZERO)
如果 TEMP ≠ 0;
PC = 地址 (NZERO)

GOTO 无条件转移

语法： [*label*] GOTO k

操作数： $0 \le k \le 1048575$

操作： k → PC<20:1>

受影响的状态位： 无

机器码：

第 1 个字（k<7:0>）	1110	1111	$k_7$kkk	kkkk$_0$
第 2 个字（k<19:8>）	1111	k_{19}kkk	kkkk	kkkk$_8$

描述： GOTO 允许无条件地转移到 2MB 存储空间中的任何地方。将 20 位值 “k” 装入 PC<20:1>。 GOTO 始终为双周期指令。

指令字： 2

指令周期： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k<7:0>，	无操作	读立即数 k<19:8>，写入 PC
无操作	无操作	无操作	无操作

示例： GOTO THERE

执行指令后
PC = 地址 (THERE)

INCF f 加 1

语法： [*label*] INCF f [,d [,a]]

操作数： $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

操作： (f)+ 1 → dest

受影响的状态位： C, DC, N, OV, Z

机器码：

0010	10da	ffff	ffff

描述： 寄存器 f 的内容加 1。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回到寄存器 f （默认情况）。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区 （默认情况）。

指令字： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	写入目标寄存器

示例： INCF CNT, 1, 0

执行指令前
CNT = 0xFF
Z = 0
C = ?
DC = ?

执行指令后
CNT = 0x00
Z = 1
C = 1
DC = 1

INCFSZ **f 加 1，为 0 则跳过**

语法： [*label*] INCFSZ f [,d [,a]]

操作数： 0 ϒ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (f) + 1 → dest，
结果为 0 则跳过

受影响的状态位： 无

机器码：

0011	11da	ffff	ffff

描述： 寄存器 f 的内容加 1。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回寄存器 f（默认情况）。
如果结果为 0，则不再执行取得的下一条指令，转而执行一条 NOP 指令，从而该指令变成双周期指令。
如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区（默认情况）。

指令字： 1

指令周期： 1(2)
注： 如果跳过一条指令，继续执行一条双字指令，则为 3 周期指令。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 “f”	处理 数据	写入 目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
无 操作	无 操作	无 操作	无 操作

如果跳过且后面跟有双 2 字指令：

Q1	Q2	Q3	Q4
无 操作	无 操作	无 操作	无 操作
无 操作	无 操作	无 操作	无 操作

示例：

```
HERE     INCFSZ   CNT, 1, 0
NZERO    :
ZERO     :
```

执行指令前
PC = 地址 (HERE)

执行指令后
CNT = CNT + 1
如果 CNT = 0，
PC = 地址 (ZERO)
如果 CNT ≠ 0，
PC = 地址 (NZERO)

INFSNZ **f 加 1，非 0 则跳过**

语法： [*label*] INFSNZ f [,d [,a]]

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (f) + 1 → dest，
结果非 0 则跳过

受影响的状态位： 无

机器码：

0100	10da	ffff	ffff

描述： 寄存器 f 的内容加 1。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回寄存器 f（默认情况）。
如果结果不为 0，则不再执行取得的下一条指令，转而执行一条 NOP 指令，从而该指令变成双周期指令。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区（默认情况）。

指令字： 1

指令周期： 1(2)
注： 如果跳过一条指令，继续执行一条双字指令，则为 3 周期指令。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 “f”	处理 数据	写入 目标寄存器

如果跳过：

Q1	Q2	Q3	Q4
无 操作	无 操作	无 操作	无 操作

如果跳过且后面跟有双字指令：

Q1	Q2	Q3	Q4
无 操作	无 操作	无 操作	无 操作
无 操作	无 操作	无 操作	无 操作

示例：

```
HERE     INFSNZ   REG, 1, 0
ZERO
NZERO
```

执行指令前
PC = 地址 (HERE)

执行指令后
REG = REG + 1
如果 REG ≠ 0，
PC = 地址 (NZERO)
如果 REG = 0，
PC = 地址 (ZERO)

IORLW 立即数和 W 进行 “或”操作

语法：	[*label*] IORLW k
操作数：	0 ≤ k ≤ 255
操作：	(W) .OR. k → W
受影响的状态位：	N, Z
机器码：	0000 | 1001 | kkkk | kkkk
描述：	W 中的内容与 8 位立即数 “k”进行 “或”操作。结果存入 W。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 “k”	处理数据	写入 W

示例： `IORLW 0x35`

执行指令前
W = 0x9A

执行指令后
W = 0xBF

IORWF W 和 f 进行 “或”操作

语法：	[*label*] IORWF f [,d [,a]]
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(W) .OR. (f) → dest
受影响的状态位：	N, Z
机器码：	0001 | 00da | ffff | ffff
描述：	W 与 f 寄存器的内容进行 “或”操作。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回寄存器 “f”（默认情况）。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区（默认情况）。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 “f”	处理 数据	写入 目标寄存器

示例： `IORWF RESULT, 0, 1`

执行指令前
RESULT = 0x13
W = 0x91

执行指令后
RESULT = 0x13
W = 0x93

LFSR　　载入 FSR

语法：　[*label*]　LFSR　f,k

操作数：　$0 \le f \le 2$
$0 \le k \le 4095$

操作：　$k \rightarrow FSRf$

受影响的状态位：　无

机器码：

1110	1110	00ff	k_{11}kkk
1111	0000	$k_7$kkk	kkkk

描述：　12 位立即数 k 装入 f 指向的文件选择寄存器。

指令字：　2

指令周期：　2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k 的 MSB	处理数据	将立即数 k 的 MSB 写入 FSRfH
译码	读立即数 k 的 LSB	处理数据	将立即数k 的 LSB 写入 FSRfL

示例：　`LFSR 2, 0x3AB`

执行指令后
FSR2H　=　0x03
FSR2L　=　0xAB

MOVF　　传送寄存器 f 的内容

语法：　[*label*]　MOVF　f [,d [,a]]

操作数：　$0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

操作：　$f \rightarrow dest$

受影响的状态位：　N, Z

机器码：

0101	00da	ffff	ffff

描述：　根据 d 的状态，将寄存器 f 的内容移入目标寄存器。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回寄存器 f （默认情况）。 f 可以是 256 字节存储区中的任何存储单元。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区 （默认情况）。

指令字：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	写入 W 寄存器

示例：　`MOVF　REG, 0, 0`

执行指令前
REG　=　0x22
W　=　0xFF

执行指令后
REG　=　0x22
W　=　0x22

MOVFF 将 f_s 的内容传送到 f_d

语法： [*label*] MOVFF f_s,f_d

操作数： $0 \le f_s \le 4095$
$0 \le f_d \le 4095$

操作： $(f_s) \rightarrow f_d$

受影响的状态位： 无

机器码：
第 1 个字 （源）
第 2 个字 （目标）

1100	ffff	ffff	$ffff_s$
1111	ffff	ffff	$ffff_d$

描述： 将源寄存器 f_s 的内容移到目标寄存器 f_d。源 f_s 可以是 4096 字节数据空间 （000h 到 FFFh）中的任何存储单元，目标 f_d 也可以是 000h 到 FFFh 中的任何存储单元。
源或目标都可以是 W （这是个有用的特殊情况）。
MOVFF 在将数据存储单元传递到外设寄存器 （如发送缓冲器或 I/O 端口）时特别有用。

MOVFF 指令中的目标寄存器不能是 PCL、 TOSU、 TOSH 或 TOSL。

注： 在使能任何中断时，不应该使用 MOVFF 指令修改中断设置。具体信息请参见第 8.0 节。

指令字： 2

指令周期： 2 (3)

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”（源）	处理数据	无操作
译码	无操作 无哑读取	无操作	写寄存器 “f”（目标）

示例： `MOVFF REG1, REG2`

执行指令前
REG1 = 0x33
REG2 = 0x11

执行指令后
REG1 = 0x33，
REG2 = 0x33

MOVLB 立即数发送到 **BSR** 的低 **4** 位

语法： [*label*] MOVLB k

操作数： $0 \le k \le 255$

操作： $k \rightarrow BSR$

受影响的状态位： 无

机器码：

0000	0001	kkkk	kkkk

描述： 将 8 位立即数 k 装入存储区选择寄存器 （BSR）。

指令字： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 k	处理数据	将立即数 k 写入 BSR

示例： `MOVLB 5`

执行指令前
BSR 寄存器 = 0x02

执行指令后
BSR 寄存器 = 0x05

MOVLW 立即数发送到 **W**

语法： [*label*] MOVLW k

操作数： $0 \leq k \leq 255$

操作： $k \rightarrow W$

受影响的状态位： 无

机器码：

0000	1110	kkkk	kkkk

描述： 将 8 位立即数 k 装入 W。

指令字： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 k	处理 数据	写入 W

示例： `MOVLW 0x5A`

执行指令后
W = 0x5A

MOVWF 将 **W** 传送到 **f**

语法： [*label*] MOVWF f [,a]

操作数： $0 \leq f \leq 255$
$a \in [0,1]$

操作： $(W) \rightarrow f$

受影响的状态位： 无

机器码：

0110	111a	ffff	ffff

描述： 将数据从 W 移到寄存器 f。f 可以是 256 字节存储区中的任何存储单元。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区 （默认情况）。

指令字： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 “f”	处理数据	写入 寄存器 “f”

示例： `MOVWF REG, 0`

执行指令前
W = 0x4F
REG = 0xFF

执行指令后
W = 0x4F
REG = 0x4F

MULLW　立即数和 W 相乘

语法：	[*label*]　MULLW　k
操作数：	0 ≤ k ≤ 255
操作：	(W) x k → PRODH:PRODL
受影响的状态位：	无
机器码：	0000 1101 kkkk kkkk
描述：	W 的内容与 8 位立即数 k 执行无符号乘法运算。 16 位乘积结果保存在 PRODH:PRODL 寄存器对中。 PRODH 包含高位字节。 W 的内容不变。 所有状态标志位都不受影响。 请注意此操作不可能发生溢出或进位。结果有可能为零，但不会被检测到。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 “k”	处理 数据	写入寄存器 PRODH: PRODL

示例：　`MULLW　0xC4`

执行指令前
- W = 0xE2
- PRODH = ?
- PRODL = ?

执行指令后
- W = 0xE2
- PRODH = 0xAD
- PRODL = 0x08

MULWF　W 与 f 相乘

语法：	[*label*]　MULWF　f [,a]
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]
操作：	(W) x (f) → PRODH:PRODL
受影响的状态位：	无
机器码：	0000 001a ffff ffff
描述：	将 W 和寄存器文件存储单元 f 中的内容执行无符号乘法运算。运算的 16 位结果保存在 PRODH:PRODL 寄存器对中。 PRODH 包含高位字节。 W 与 f 的内容都不变。 所有状态标志位都不受影响。 请注意此操作不可能发生溢出或进位。结果有可能为零，但不会被检测到。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区 （默认情况）。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 “f”	处理 数据	写入寄存器 PRODH: PRODL

示例：　`MULWF　REG, 1`

执行指令前
- W = 0xC4
- REG = 0xB5
- PRODH = ?
- PRODL = ?

执行指令后
- W = 0xC4
- REG = 0xB5
- PRODH = 0x8A
- PRODL = 0x94

NEGF 对 f 求补

语法：	[*label*] NEGF f [,a]
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]
操作：	$(\overline{f}) + 1 \rightarrow f$
受影响的状态位：	N, OV, C, DC, Z
机器码：	`0110` `110a` `ffff` `ffff`
描述：	用二进制补码对单元 f 求补。结果保存在数据存储单元 “f” 中。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 “f”	处理 数据	写入 寄存器 “f”

示例： `NEGF REG, 1`

执行指令前
REG = `0011 1010` [0x3A]

执行指令后
REG = `1100 0110` [0xC6]

NOP 无操作

语法：	[*label*] NOP
操作数：	无
操作：	无操作
受影响的状态位：	无
机器码：	`0000` `0000` `0000` `0000` `1111` `xxxx` `xxxx` `xxxx`
描述：	无操作。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	无 操作	无 操作	无 操作

示例：

无

POP　将返回堆栈顶部的内容弹出

语法：	[*label*]　POP
操作数：	无
操作：	(TOS) → 位地址
受影响的状态位：	无
机器码：	0000 0000 0000 0110
描述：	从返回堆栈取出 TOS 值并丢弃。前一个压入返回堆栈的值随后成为 TOS 值。 此指令可以让用户正确管理返回堆栈以组成软件堆栈。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	无操作	弹出 TOS 值	无操作

示例：

```
POP
GOTO      NEW
```

执行指令前
- TOS = 0031A2h
- 堆栈 （向下 1 级） = 014332h

执行指令后
- TOS = 014332h
- PC = NEW

PUSH　将内容压入返回堆栈的顶部

语法：	[*label*]　PUSH
操作数：	无
操作：	(PC+2) → TOS
受影响的状态位：	无
机器码：	0000 0000 0000 0101
描述：	PC+2 被压入返回堆栈的顶部。原先的 TOS 值压入堆栈的下一层。 此指令允许通过修改 TOS 来实现软件堆栈，然后将其压入返回堆栈。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	将 PC+2 压入返回堆栈	无操作	无操作

示例：

```
PUSH
```

执行指令前
- TOS = 00345Ah
- PC = 000124h

执行指令后
- PC = 000126h
- TOS = 000126h
- 堆栈 （向下 1 级） = 00345Ah

RCALL 相对调用

语法： [*label*] RCALL n

操作数： $-1024 \le n \le 1023$

操作： (PC) + 2 → TOS,
(PC) + 2 + 2n → PC

受影响的状态位： 无

机器码：

1101	1nnn	nnnn	nnnn

描述： 从当前单元跳转 （最多 1K 范围） 来调用子程序。首先，返回地址 （PC+2）被压入堆栈。然后，将 PC 加上二进制补码 “2n”。因为 PC 要先递增才能取下一条指令，因此新地址将为 PC+2+2n。这是一条双周期的指令。

指令字： 1

指令周期： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 “n” 将 PC 压入堆栈	处理数据	写入 PC
无操作	无操作	无操作	无操作

示例： HERE RCALL Jump

执行指令前
PC = 地址 (HERE)

执行指令后
PC = 地址 (Jump)
TOS = 地址 (HERE+2)

RESET 复位

语法： [*label*] RESET

操作数： 无

操作： 将所有受 $\overline{MCLR}$ 复位影响的寄存器和标志复位。

受影响的状态位： 全部

机器码：

0000	0000	1111	1111

描述： 此指令可用于在软件中执行 $\overline{MCLR}$ 复位。

指令字： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	开始复位	无操作	无操作

示例： RESET

执行指令后
寄存器 = 复位值
标志位 * = 复位值

RETFIE 从中断返回

语法：	[*label*] RETFIE [s]
操作数：	s ∈ [0,1]
操作：	(TOS) → PC, 1 → GIE/GIEH 或 PEIE/GIEL, 如果 s = 1 (WS) → W, (STATUSS) → STATUS， (BSRS) → BSR， PCLATU 和 PCLATH 不变。
受影响的状态位：	GIE/GIEH， PEIE/GIEL。
机器码：	0000 0000 0001 000s
描述：	从中断返回。执行出栈操作，将栈顶 （Top-of-Stack， TOS）单元内容装入 PC。通过将高 / 低优先级全局中断使能位置 “1”，可以使能中断。如果 s 为 1，将影子寄存器 WS、STATUSS 和 BSRS 的内容装入对应的寄存器 W、 STATUS 和 BSR。如果 s 非 0，则不会更新这些寄存器 （默认情况）。
指令字：	1
指令周期：	2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	无 操作	无 操作	从堆栈弹出 PC 将 GIEH 或 GIEL 置 1
无 操作	无 操作	无 操作	无 操作

示例： `RETFIE 1`

中断后

PC	=	TOS
W	=	WS
BSR	=	BSRS
STATUS	=	STATUSS
GIE/GIEH， PEIE/GIEL	=	1

RETLW 向 W 寄存器返回立即数

语法：	[*label*] RETLW k
操作数：	0 ≤ k ≤ 255
操作：	k → W, (TOS) → PC, PCLATU 和 PCLATH 不变
受影响的状态位：	无
机器码：	0000 1100 kkkk kkkk
描述：	将 8 位立即数 k 装入 W。将栈顶单元内容 （返回地址）装入程序计数器。高位地址锁存器 （PCLATH）保持不变。
指令字：	1
指令周期：	2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 立即数 “k”	处理 数据	从堆栈弹出 PC， 写入 W
无 操作	无 操作	无 操作	无 操作

示例：

```
    CALL TABLE ; W contains table
               ; offset value
               ; W now has
               ; table value
   :
TABLE
    ADDWF PCL  ; W = offset
    RETLW k0   ; Begin table
    RETLW k1   ;
   :
   :
    RETLW kn   ; End of table
```

执行指令前
W = 0x07

执行指令后
W = 值 kn

RETURN 从子程序返回

语法：	[*label*] RETURN [s]
操作数：	s ∈ [0,1]
操作：	(TOS) → PC, 如果 s = 1 (WS) → W, (STATUSS) → STATUS， (BSRS) → BSR， PCLATU 和 PCLATH 不变
受影响的状态位：	无
机器码：	0000 0000 0001 001s
描述：	从子程序返回。弹出堆栈中的数据，并将栈顶 （TOS）单元内容装入程序计数器。如果 s 为 1，将影子寄存器 WS、STATUSS 和 BSRS 的内容被装入对应的寄存器 W、STATUS 和 BSR。如果 s 为 0，则不会更新这些寄存器 （默认情况）。
指令字：	1
指令周期：	2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	无 操作	处理 数据	从堆栈弹出 PC
无 操作	无 操作	无 操作	无 操作

示例： `RETURN`

中断后
PC = TOS

RLCF 带进位循环左移 f

语法：	[*label*] RLCF f [,d [,a]]
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(f<n>)→ dest<n+1>， (f<7>)→ C， (C) → dest<0>
受影响的状态位：	C, N, Z
机器码：	0011 01da ffff ffff
描述：	寄存器 “f” 的内容带进位标志循环左移 1 位。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回到 f 寄存器 （默认情况）。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区 （默认情况）。
	C ← 寄存器 f
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 “f”	处理 数据	写入 目标寄存器

示例： `RLCF REG, 0, 0`

执行指令前
REG = 1110 0110
C = 0

执行指令后
REG = 1110 0110
W = 1100 1100
C = 1

RLNCF 循环左移 f （无进位）

语法： [*label*] RLNCF f [,d [,a]]

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (f<n>) → dest<n+1>，
(f<7>) →dest<0>

受影响的状态位： N, Z

机器码：

0100	01da	ffff	ffff

描述： 寄存器 f 的内容循环左移 1 位。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回到 f 寄存器 （默认情况）。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区 （默认情况）。

寄存器 f

指令字： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	写入目标寄存器

示例： `RLNCF REG, 1, 0`

执行指令前
REG = 1010 1011

执行指令后
REG = 0101 0111

RRCF 带进位循环右移 f

语法： [*label*] RRCF f [,d [,a]]

操作数： 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

操作： (f<n>) → dest<n-1>，
(f<0>) → C，
(C) → dest<7>

受影响的状态位： C, N, Z

机器码：

0011	00da	ffff	ffff

描述： 寄存器 f 的内容带进位标志循环右移 1 位。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回到寄存器 f （默认情况）。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区 （默认情况）。

C 寄存器 f

指令字： 1

指令周期： 1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	写入目标寄存器

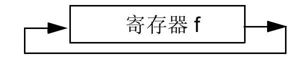
示例： `RRCF REG, 0, 0`

执行指令前
REG = 1110 0110
C = 0

执行指令后
REG = 1110 0110
W = 0111 0011
C = 0

RRNCF 循环右移 **f** （无进位）

语法：	[*label*] RRNCF f [,d [,a]]
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(f<n>) → dest<n-1>， (f<0>) → dest<7>
受影响的状态位：	N, Z
机器码：	0100 | 00da | ffff | ffff
描述：	寄存器 f 的内容循环右移 1 位。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回到寄存器 f （默认情况）。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区 （默认情况）。



指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 “f”	处理 数据	写入 目标寄存器

示例 1： `RRNCF REG, 1, 0`

执行指令前
REG = 1101 0111

执行指令后
REG = 1110 1011

示例 2： `RRNCF REG, 0, 0`

执行指令前
W = ?
REG = 1101 0111

执行指令后
W = 1110 1011
REG = 1101 0111

SETF 置位 **f**

语法：	[*label*] SETF f [,a]
操作数：	0 ≤ f ≤ 255 a ∈ [0,1]
操作：	FFh → f
受影响的状态位：	无
机器码：	0110 | 100a | ffff | ffff
描述：	将指定寄存器的内容设为 FFh. 如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区 （默认情况）。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 “f”	处理 数据	写 寄存器 “f”

示例： `SETF REG,1`

执行指令前
REG = 0x5A

执行指令后
REG = 0xFF

SLEEP 进入休眠模式

语法：	[*label*] SLEEP
操作数：	无
操作：	00h → WDT, 0 → WDT 后分频器， 1 → $\overline{TO}$, 0 → $\overline{PD}$
受影响的状态位：	$\overline{TO}$, $\overline{PD}$
机器码：	`0000 0000 0000 0011`
描述：	掉电状态位（$\overline{PD}$）清零。超时状态位（$\overline{TO}$）置 1。看门狗定时器及其后分频器清零。 处理器进入休眠模式，振荡器停振。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	无操作	处理数据	进入休眠

示例： `SLEEP`

执行指令前
- $\overline{TO}$ = ?
- $\overline{PD}$ = ?

执行指令后
- $\overline{TO}$ = 1 †
- $\overline{PD}$ = 0

† 如果 WDT 唤醒处理器，则此位将清零。

SUBFWB W 减去 f 和借位

语法：	[*label*] SUBFWB f [,d [,a]]
操作数：	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
操作：	$(W) - (f) - (\overline{C}) \rightarrow dest$
受影响的状态位：	N, OV, C, DC, Z
机器码：	`0101 01da ffff ffff`
描述：	W 减去 f 寄存器和进位标志（借位）（采用二进制补码方法）。如果 d 为 0，结果存入 W。如果 d 为 1，结果存入 f 寄存器（默认情况）。如果 ‘a’ = 0，则忽略 BSR 值而选择存取存储区。如果 ‘a’ = 1，则按照 BSR 值选择存储区（默认情况）。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	写入目标寄存器

示例 1： `SUBFWB REG, 1, 0`

执行指令前
- REG = 3
- W = 2
- C = 1

执行指令后
- REG = FF
- W = 2
- C = 0
- Z = 0
- N = 1 ; result is negative

示例 2： `SUBFWB REG, 0, 0`

执行指令前
- REG = 2
- W = 5
- C = 1

执行指令后
- REG = 2
- W = 3
- C = 1
- Z = 0
- N = 0 ; result is positive

例 3： `SUBFWB REG, 1, 0`

执行指令前
- REG = 1
- W = 2
- C = 0

执行指令后
- REG = 0
- W = 2
- C = 1
- Z = 1 ; result is zero
- N = 0

SUBLW 立即数减去 W

语法：	[*label*] SUBLW k
操作数：	0 ≤ k ≤ 255
操作：	k − (W) → W
受影响的状态位：	N, OV, C, DC, Z
机器码：	`0000` `1000` `kkkk` `kkkk`
描述：	8 位立即数 k 减去 W 寄存器的内容。结果保入 W。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 “k”	处理数据	写入 W

示例 1： `SUBLW 0x02`

执行指令前
```
W = 1
C = ?
```
执行指令后
```
W = 1
C = 1    ; result is positive
Z = 0
N = 0
```

示例 2： `SUBLW 0x02`

执行指令前
```
W = 2
C = ?
```
执行指令后
```
W = 0
C = 1    ; result is zero
Z = 1
N = 0
```

示例 3： `SUBLW 0x02`

执行指令前
```
W = 3
C = ?
```
执行指令后
```
W = FF   ; (2's complement)
C = 0    ; result is negative
Z = 0
N = 1
```

SUBWF f 减去 W

语法：	[*label*] SUBWF f [,d [,a]]
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(f) − (W) → dest
受影响的状态位：	N, OV, C, DC, Z
机器码：	`0101` `11da` `ffff` `ffff`
描述：	寄存器 f 的内容减去 W （采用二进制补码方法）。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回 f 寄存器 （默认情况）。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区 （默认情况）。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	写入目标寄存器

示例 1： `SUBWF REG, 1, 0`

执行指令前
```
REG = 3
W   = 2
C   = ?
```
执行指令后
```
REG = 1
W   = 2
C   = 1    ; result is positive
Z   = 0
N   = 0
```

示例 2： `SUBWF REG, 0, 0`

执行指令前
```
REG = 2
W   = 2
C   = ?
```
执行指令后
```
REG = 2
W   = 0
C   = 1    ; result is zero
Z   = 1
N   = 0
```

示例 3： `SUBWF REG, 1, 0`

执行指令前
```
REG = 1
W   = 2
C   = ?
```
执行指令后
```
REG = FFh  ;(2's complement)
W   = 2
C   = 0    ; result is negative
Z   = 0
N   = 1
```

SUBWFB　f 减去 W 和借位

语法：[*label*] SUBWFB f [,d [,a]]

操作数：$0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

操作：$(f) - (W) - (\overline{C}) \rightarrow dest$

受影响的状态位：N, OV, C, DC, Z

机器码：

0101	10da	ffff	ffff

描述：f 寄存器的内容减去 W 寄存器内容和进位标志 （借位）（采用二进制补码方法）。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回 f 寄存器（默认情况）。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区（默认情况）。

指令字：1

指令周期：1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	写入目标寄存器

示例 1：SUBWFB REG, 1, 0

执行指令前
REG = 0x19 (0001 1001)
W = 0x0D (0000 1101)
C = 1

执行指令后
REG = 0x0C (0000 1011)
W = 0x0D (0000 1101)
C = 1
Z = 0
N = 0 ; result is positive

示例 2：SUBWFB REG, 0, 0

执行指令前
REG = 0x1B (0001 1011)
W = 0x1A (0001 1010)
C = 0

执行指令后
REG = 0x1B (0001 1011)
W = 0x00
C = 1
Z = 1 ; result is zero
N = 0

例 3：SUBWFB REG, 1, 0

执行指令前
REG = 0x03 (0000 0011)
W = 0x0E (0000 1110)
C = 1

执行指令后
REG = 0xF5 (1111 0101)
; [2's comp]
W = 0x0E (0000 1110)
C = 0
Z = 0
N = 1 ; result is negative

SWAPF　f 半字节交换

语法：[*label*] SWAPF f [,d [,a]]

操作数：$0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

操作：$(f<3:0>) \rightarrow dest<7:4>$，
$(f<7:4>) \rightarrow dest<3:0>$

受影响的状态位：无

机器码：

0011	10da	ffff	ffff

描述：互换 f 寄存器的高 4 位字节和低 4 位字节。如果 d 为 0，结果存入 W。如果 d 为 1，结果存入 f 寄存器（默认情况）。如果 a 位 0，则忽略 BSR 值而选择存取存储区。如果 a 位 1，则按照 BSR 值选择存储区（默认情况）。

指令字：1

指令周期：1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	写入目标寄存器

示例：SWAPF REG, 1, 0

执行指令前
REG = 0x53

执行指令后
REG = 0x35

TBLRD 读表

语法：	[*label*] TBLRD (*; *+; *-; +*)
操作数：	无
操作：	对于 TBLRD *， (Prog Mem(TBLPTR)) → TABLAT ； TBLPTR 不变； 对于 TBLRD *+， (Prog Mem(TBLPTR)) → TABLAT ； (TBLPTR) +1 → TBLPTR ； 对于 TBLRD *-， (Prog Mem(TBLPTR)) → TABLAT ； (TBLPTR) -1 → TBLPTR ； 对于 TBLRD +*， (TBLPTR) +1 → TBLPTR ； (Prog Mem(TBLPTR)) → TABLAT ；
受影响的状态位：	无

机器码：

0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*

描述： 本指令用于读取程序存储器 （PM）的内容。使用表指针 （TBLPTR）对程序存储器进行寻址。
TBLPTR （一个 21 位的指针）指向程序存储器的每个字节。 TBLPTR 寻址范围为 2MB。

TBLPTR[0] = 0: 程序存储器字的最低有效字节

TBLPTR[0] = 1: 程序存储器字的最高有效字节

TBLRD 指令可以如下修改 TBLPTR 的值：

- 不变
- 后递增
- 后递减
- 前递增

指令字： 1

指令周期： 2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	无操作	无操作	无操作
无操作	无操作（读程序存储器）	无操作	无操作（写 TABLAT）

TBLRD 读表 （续）

示例 1： `TBLRD *+ ;`

执行指令前

TABLAT	=	0x55
TBLPTR	=	0x00A356
MEMORY(0x00A356)	=	0x34

执行指令后

TABLAT	=	0x34
TBLPTR	=	0x00A357

示例 2： `TBLRD +* ;`

执行指令前

TABLAT	=	0xAA
TBLPTR	=	0x01A357
MEMORY(0x01A357)	=	0x12
MEMORY(0x01A358)	=	0x34

执行指令后

TABLAT	=	0x34
TBLPTR	=	0x01A358

TBLWT 写表

语法：	[*label*] TBLWT (*; *+; *-; +*)
操作数：	无
操作：	对于 TBLWT*， (TABLAT) → 保持寄存器， TBLPTR 不变； 对于 TBLWT*+， (TABLAT) → 保持寄存器， (TBLPTR) +1 → TBLPTR ； 对于 TBLWT*-， (TABLAT) → 保持寄存器， (TBLPTR) -1 → TBLPTR ； 对于 TBLWT+*， (TBLPTR)+1 → TBLPTR ； (TABLAT) → 保持寄存器，
受影响的状态位：	无
机器码：	0000 | 0000 | 0000 | 11nn nn=0 * =1 *+ =2 *- =3 +*
描述：	本指令使用 TBLPTR 的 3 个 LSb 来决定要将 TABLAT 数据写入 8 个保持寄存器中的哪一个。 8 个保持寄存器用于对程序存储器 （PM）的内容编程。有关写入闪存存储器的信息，参见第 5.0 节。 TBLPTR （一个 21 位指针）指向程序存储器的每个字节。TBLPTR 寻址范围为 2MB。 TBLPTR 的 LSb 决定要访问程序存储单元的哪个字节。 TBLPTR[0] = 0: 程序存储器字的最低有效字节 TBLPTR[0] = 1: 程序存储器字的最高有效字节 TBLWT 指令可以如下修改 TBLPTR 的值： • 不变 • 后递增 • 后递减 • 前递增
指令字：	1
指令周期：	2

Q 周期操作：

Q1	Q2	Q3	Q4
译码	无操作	无操作	无操作
无操作	无操作（读 TABLAT）	无操作	无操作（写保持寄存器或存储器）

TBLWT 写表 （续）

示例 1： `TBLWT *+;`

执行指令前

TABLAT	=	0x55
TBLPTR	=	0x00A356
保持寄存器 (0x00A356)	=	0xFF

指令执行后 （表写入完成）

TABLAT	=	0x55
TBLPTR	=	0x00A357
保持寄存器 (0x00A356)	=	0x55

示例 2： `TBLWT +*;`

执行指令前

TABLAT	=	0x34
TBLPTR	=	0x01389A
保持寄存器 (0x01389A)	=	0xFF
保持寄存器 (0x01389B)	=	0xFF

指令执行后 （表写入完成）

TABLAT	=	0x34
TBLPTR	=	0x01389B
保持寄存器 (0x01389A)	=	0xFF
保持寄存器 (0x01389B)	=	0x34

TSTFSZ　测试 f，为 0 时跳过

语法：　[*label*]　TSTFSZ　f [,a]

操作数：　$0 \le f \le 255$
$a \in [0,1]$

操作：　如果 f = 0 则跳过

受影响的状态位：　无

机器码：

0110	011a	ffff	ffff

描述：　如果 f 为 0，将不执行在当前指令执行时所取的下一条指令，转而执行一条 NOP 指令，从而使该指令变成双周期指令。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区（默认情况）。

指令字：　1

指令周期：　1(2)
注：如果跳过一条指令，然后继续执行双字指令，则为 3 周期指令。

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读寄存器 “f”	处理数据	无操作

如果跳过：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作

如果跳过后面执行双字指令：

Q1	Q2	Q3	Q4
无操作	无操作	无操作	无操作
无操作	无操作	无操作	无操作

示例：

```
HERE     TSTFSZ  CNT, 1
NZERO    :
ZERO     :
```

执行指令前
PC = 地址 (HERE)

执行指令后
如果 CNT　=　0x00，
PC　=　地址 (ZERO)
如果 CNT　≠　0x00，
PC　=　地址 （(NZERO)

XORLW　立即数和 W 进行 “异或”操作

语法：　[*label*] XORLW　k

操作数：　$0 \le k \le 255$

操作：　(W) .XOR. k → W

受影响的状态位：　N, Z

机器码：

0000	1010	kkkk	kkkk

描述：　W 的内容与 8 位立即数 k 进行 “异或”操作。结果存入 W。

指令字：　1

指令周期：　1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读立即数 “k”	处理数据	写 W

示例：　XORLW　0xAF

执行指令前
W　=　0xB5

执行指令后
W　=　0x1A

XORWF	W 和 f 进行 “异或”操作
语法：	[*label*] XORWF f [,d [,a]]
操作数：	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
操作：	(W) .XOR.(f) → dest
受影响的状态位：	N, Z
机器码：	`0001` `10da` `ffff` `ffff`
描述：	将 W 的内容与寄存器 f 的内容进行“异或”操作。如果 d 为 0，结果存入 W。如果 d 为 1，结果存回 f 寄存器 （默认情况）。如果 a 为 0，则忽略 BSR 值而选择存取存储区。如果 a 为 1，则按照 BSR 值选择存储区 （默认情况）。
指令字：	1
指令周期：	1

Q 周期操作：

Q1	Q2	Q3	Q4
译码	读 寄存器 “f”	处理 数据	写入 目标寄存器

示例： `XORWF REG, 1, 0`

执行指令前
REG = 0xAF
W = 0xB5

执行指令后
REG = 0x1A
W = 0xB5

21.0 开发支持

一系列硬件及软件开发工具对 PICmicro® 单片机提供支持：

- 集成开发环境
  - MPLAB® IDE 软件
- 汇编器 / 编译器 / 链接器
  - MPASM™ 汇编器
  - MPLAB C17 和 MPLAB C18 C 编译器
  - MPLINK™ 目标链接器 / MPLIB™ 目标库管理器
  - MPLAB C30 C 编译器
  - MPLAB ASM30 汇编器 / 链接器 / 库
- 模拟器
  - MPLAB SIM 软件模拟器
  - MPLAB dsPIC30 软件模拟器
- 仿真器
  - MPLAB ICE 2000 在线仿真器
  - MPLAB ICE 4000 在线仿真器
- 在线调试器
  - MPLAB ICD 2
- 器件编程器
  - PRO MATE® II 通用器件编程器
  - PICSTART® Plus 开发编程器
  - MPLAB PM3 器件编程器
- 低成本演示板
  - PICDEM™ 1 演示板
  - PICDEM.net™ 演示板
  - PICDEM 2 Plus 演示板
  - PICDEM 3 演示板
  - PICDEM 4 演示板
  - PICDEM 17 演示板
  - PICDEM 18R 演示板
  - PICDEM LIN 演示板
  - PICDEM USB 演示板
- 评估工具包
  - KEELOQ® 评估和编程工具
  - PICDEM MSC
  - microID® 开发工具包
  - CAN
  - PowerSmart® 开发工具包
  - 模拟

21.1 MPLAB 集成开发环境软件

MPLAB IDE 软件为 8/16 位单片机市场提供了前所未有的易于使用的软件开发平台。 MPLAB IDE 是基于 Windows® 平台的应用软件，包括：

- 调试工具接口
  - 模拟器
  - 编程器 （单独销售）
  - 仿真器 （单独销售）
  - 在线调试器 （单独销售）
- 具有彩色上下文代码显示的全功能编辑器
- 多项目管理器
- 内容可直接编辑的可定制式数据窗口
- 高级源代码调试
- 鼠标停留在变量上进行查看的功能
- 丰富的在线帮助

MPLAB IDE 可以让您：

- 编辑源文件 （汇编语言或 C 语言）
- 点击一次即可完成汇编 （或编译）并将代码下载到 PICmicro 仿真器和模拟器工具中 （自动更新所有项目信息）
- 可使用如下各项进行调试：
  - 源文件 （汇编语言或 C 语言）
  - 混合汇编语言和 C 语言
  - 机器码

MPLAB IDE 在单个开发范例中支持使用多种调试工具，包括从成本效益高的模拟器到低成本的在线调试器，再到全功能的仿真器。这样缩短了用户升级到更加灵活而功能更强大的工具时的学习时间。

21.2 MPASM 汇编器

MPASM 汇编器是全功能通用宏汇编器，适用于所有的 PICmicro MCU。

MPASM 汇编器可生成用于 MPLINK 目标链接器的可重定位目标文件、Intel® 标准 HEX 文件、详细描述存储器使用状况和符号参考的 MAP 文件、包含源代码行及生成机器码的绝对 LST 文件以及用于调试的 COFF 文件。

MPASM 汇编器具有如下特征：

- 集成在 MPLAB IDE 项目中
- 用户定义的宏可简化汇编代码
- 对多用途源文件进行条件汇编
- 允许完全控制汇编过程的指令

21.3 MPLAB C17 和 MPLAB C18 C 编译器

MPLAB C17 和 MPLAB C18 代码开发系统是完全的 ANSI C 编译器，分别适用于 Microchip 的 PIC17CXXX 和 PIC18CXXX 系列单片机。这些编译器可提供其他编译器并不具备的强大的集成功能和出众的代码优化能力，且使用方便。

为便于源代码调试，编译器提供了针对 MPLAB IDE 调试器的优化符号信息。

21.4 MPLINK 目标链接器 /MPLIB 目标库管理器

MPLINK 目标链接器包含了由 MPASM 汇编器、MPLAB C17 和 MPLAB C18 C 编译器产生的可重定位目标。通过使用链接器脚本中的指令，它还可链接预编译库中的可重定位目标。

MPLIB目标库管理器管理预编译代码库文件的创建和修改。当从源文件调用库中的一段子程序时，只有包含此子程序的模块被链接到应用中。这样可使大型库在许多不同应用中被高效地利用。

目标链接器 / 库管理器具有如下特征：

- 高效地连接单个的库而不是许多小文件
- 通过将相关的模块组合在一起来增强代码的可维护性
- 只要列出、替换、删除和抽取模块，便可灵活地创建库

21.5 MPLAB C30 C 编译器

MPLAB C30 C 编译器是全功能符合 ANSI 标准的优化编译器，它能将标准 ANSI C 程序转变成 dsPIC30F 汇编语言源代码。该编译器还支持许多命令行选项和语言扩展，以充分利用 dsPIC30F 器件的硬件功能，同时满足编译器代码发生器较高的控制要求。

MPLAB C30 附带了一个完整的 ANSI C 标准库。所有库函数已经过验证且符合 ANSI C 库标准。该库包括执行字符串操作、动态存储器分配、数据转换、时间校准等函数以及数学函数（三角函数、指数函数和双曲线函数）。该编译器提供使用 MPLAB IDE 进行高级源代码调试所用的符号信息。

21.6 MPLAB ASM30 汇编器、链接器和库管理器

MPLAB ASM30 汇编器为 dsPIC30F 器件提供转换自符号汇编语言的可重定位机器码。 MPLAB C30 编译器使用该汇编器生成目标文件。汇编器产生可重定位目标文件之后，可将这些目标文件存档，或与其他可重定位目标文件和存档链接以生成可执行文件。该汇编器有如下显著特征：

- 支持整个 dsPIC30F 指令集
- 支持定点数据和浮点数据
- 命令行界面
- 丰富的指令集
- 灵活的宏语言
- MPLAB IDE 兼容性

21.7 MPLAB SIM 软件模拟器

MPLAB SIM 软件模拟器在指令级对 PICmicro 系列单片机进行模拟，使得用户可以在 PC 主机的环境下进行代码开发。对于任何给定的指令，用户均可对数据区进行检查或修改，并通过文件或用户自定义的按键来激励任意引脚。指令的执行可采用单步、运行到断点或跟踪模式。

MPLAB SIM 模拟器完全支持使用 MPLAB C17 和 MPLAB C18 C 编译器以及MPASM汇编器的符号调试。该软件模拟器可用于在实验室环境外灵活地开发和调试代码，是一款完美且经济的软件开发工具。

21.8 MPLAB SIM30 软件模拟器

MPLAB SIM30 模拟器在指令级对 dsPIC30F 系列单片机进行模拟，使得用户可以在 PC 主机的环境下进行代码开发。对于任何给定的指令，用户可对数据区域进行检查或修改，并通过文件或用户自定义的按键来激励任意引脚。

MPLAB SIM30 模拟器完全支持使用 MPLAB C30 C 编译器和 MPLAB ASM30 汇编器的符号调试。该模拟器可运行在命令行模式实现自动批处理任务，也可运行在 MPLAB IDE 中。此高速模拟器是为调试、分析及优化时间密集型 DSP 程序而设计的。

21.9 MPLAB ICE 2000 高性能通用在线仿真器

MPLAB ICE 2000 通用在线仿真器旨在为产品开发工程师提供一整套用于 PICmicro 单片机的设计工具。MPLAB ICE 2000 在线仿真器的软件控制由 MPLAB 集成开发环境平台提供，它允许在单一环境下进行编辑、编译、下载以及源代码调试。

MPLAB ICE 2000 是全功能仿真器系统，它具有增强的跟踪、触发和数据监控功能。处理器模块可插拔，使系统可轻松进行重新配置以适应各种不同处理器的仿真需要。MPLAB ICE 在线仿真器的通用架构允许对其进行扩展以支持新的 PICmicro 单片机。

MPLAB ICE 2000 在线仿真器系统设计为一款实时仿真系统，该仿真系统具备通常只有昂贵的开发工具中才有的高级功能。选择 PC 平台和 Microsoft® Windows 32 位操作系统可使这些功能在一个简单而统一的应用中得到很好的利用。

21.10 MPLAB ICE 4000 高性能通用在线仿真器

MPLAB ICE 4000 通用在线仿真器旨在为产品开发工程师提供一整套用于高端 PICmicro 单片机的设计工具。MPLAB ICE 在线仿真器的软件控制由 MPLAB 集成开发环境平台提供，它允许在单一环境下进行编辑、编译、下载以及源代码调试。

MPLAB ICE 4000 是高级的仿真系统，除具备 MPLAB ICE 2000 的所有功能外，它还增加了适用于 dsPIC30F 和 PIC18XXXX 器件的仿真存储容量以及高速性能。该仿真器的先进特性包括复杂触发和定时功能，高达 2 Mb 的仿真存储容量以及实时变量监视功能。

MPLAB ICE 4000 在线仿真系统设计为一款实时仿真系统，该仿真系统具备通常只有在更加昂贵的开发工具中才有的高级功能。选择 PC 平台和 Microsoft Windows 32 位操作系统可使这些功能在一个简单而统一的应用程序中得以很好的利用。

21.11 MPLAB ICD 2 在线调试器

Microchip 的在线调试器 MPLAB ICD 2 是一款功能强大而成本低廉的运行时开发工具，通过 RS-232 或高速 USB 接口与 PC 主机相连。该工具基于闪存 PICmicro MCU，可用于开发本系列及其他 PICmicro 单片机。MPLAB ICD 2 使用了闪存器件中内建的在线调试功能。该功能结合 Microchip 的在线串行编程（In-Circuit Serial Programming™，ICSP™）协议，可在 MPLAB 集成开发环境的图形用户界面上提供成本效益很高的在线闪存调试。这使设计人员可通过设置断点、单步运行以及对变量、CPU 状态以及外设寄存器进行监视的方法实现源代码的开发和调试。其全速运行特性可对硬件和应用进行实时测试。MPLAB ICD 2 还可用作某些 PICmicro 器件的开发编程器。

21.12 PRO MATE II 通用器件编程器

PRO MATE II 是一款通用的、符合 CE 规范的器件编程器，其可编程电压设置在 VDDMIN 和 VDDMAX 之间时可靠性最高。它有一个 LCD 显示器用来显示指令和错误信息，以及一个支持各种封装类型的可拆卸模块化插槽装置。在单机模式下，PRO MATE II 器件编程器不必与 PC 相连即可对 PICmicro 器件进行读取、验证和编程。在该模式下它还可设置代码保护。

21.13 MPLAB PM3 器件编程器

MPLAB PM3 是一款通用的、符合 CE 规范的器件编程器，其可编程电压设置在 VDDMIN 和 VDDMAX 之间时可靠性最高。它有一个用来显示菜单和错误信息的大 LCD 显示器（128 x 64），以及一个支持各种封装类型的可拆卸模块化插槽装置。编程器标准配置中带有一根 ICSP™ 电缆。在单机模式下，MPLAB PM3 器件编程器不必与 PC 相连即可对 PICmicro 器件进行读取、验证和编程。在该模式下它还可设置代码保护。MPLAB PM3 通过 RS-232 或 USB 电缆连接到 PC 主机上。MPLAB PM3 具备高速通信能力以及优化算法，可对存储器很大的器件进行快速编程，它还采用 SD/MMC 卡用作文件存储及数据安全应用。

21.14 PICSTART Plus 开发编程器

PICSTART Plus开发编程器是一款易于使用而成本低廉的原型编程器。它通过 COM （RS-232）端口与 PC 相连。 MPLAB 集成开发环境软件使得该编程器的使用简便、高效。 PICSTART Plus 开发编程器支持大部分 PICmicro 器件，其引脚数最多可达 40 个。引脚数更多的器件，如 PIC16C92X 和 PIC17C76X，可通过连接一个转接插槽来获得支持。PICSTART Plus 开发编程器符合 CE 规范。

21.15 PICDEM 1 PICmicro 演示板

PICDEM 1 演示板可以演示 PIC16C5X （PIC16C54 到 PIC16C58A）、PIC16C61、PIC16C62X、PIC16C71、PIC16C8X、PIC17C42、PIC17C43 和 PIC17C44 单片机的功能。它包含运行基本演示程序所必需的软硬件。借助于 PRO MATE II 器件编程器或 PICSTART Plus 开发编程器，用户可对随 PICDEM 1 演示板一起提供的单片机样片编程。可将 PICDEM 1 演示板与 MPLAB ICE 在线仿真器相连，进行测试。演示板所提供的实验布线区可供用户添加应用元件来扩展电路。其他功能部件包括一个RS-232接口、一个用于仿真模拟输入的电位计、按钮开关以及 8 个 LED。

21.16 PICDEM.net 因特网 / 以太网演示板

PICDEM.net 演示板是一块使用 PIC18F452 单片机和 TCP/IP 固件的因特网 / 以太网演示板。该演示板支持所有符合 PIC16F877 或 PIC18C452 标准引脚排列的 40 引脚DIP 器件。该工具包具备使用方便的TCP/IP 堆栈、HTML 网络服务器、一个用于 Xmodem 下载至网页的 24L256 串行 EEPROM、 ICSP/MPLAB ICD 2 接口 连接器、一个以太网接口、 RS-232 接口以及一个 16 x 2 LCD 显示器。还包括 Jeremy Bentham 所著的 *"TCP/IP Lean, Web Servers for Embedded Systems"*一书及配套光盘。

21.17 PICDEM 2 Plus 演示板

PICDEM 2 Plus 演示板支持多种 18、 28 和 40 引脚的单片机，包括 PIC16F87X 和 PIC18FXX2 器件。该演示板包含了运行基本演示程序所需的软硬件。借助于PRO MATE II 器件编程器、PICSTART Plus 开发编程器或带有通用编程器适配器的 MPLAB ICD 2，用户可对随 PICDEM 2 演示板一起提供的单片机样片编程。MPLAB ICD 2 和 MPLAB ICE 在线仿真器也可以与 PICDEM 2 演示板一起使用，进行固件测试。演示板所提供的实验布线区可用来添加应用元件以扩展电路。其他功能部件包括一个 RS-232 接口、2 x 16 LCD 显示器、一个压电扬声器、一个板上温度传感器、 4 个 LED 以及 PIC18F452 和 PIC16F877 闪存单片机样片。

21.18 PICDEM 3 PIC16C92X 演示板

PICDEM 3 演示板支持 PLCC 封装形式的 PIC16C923 和PIC16C924。它包含运行基本演示程序所必需的软硬件。

21.19 PICDEM 4 8/14/18 引脚演示板

PICDEM 4 可用于演示 8、14、18 引脚 PIC16XXXX 和 PIC18XXXX MCU 的功能，包括 PIC16F818/819、PIC16F87/88、PIC16F62XA 和 PIC18F1320 单片机系列。 PICDEM 4 旨在显示这些低引脚数器件的许多功能，包括 LIN 和采用 ECCP 的电机控制功能。该演示板为低功耗操作特别提供了一些装置，如超级电容电路以及跳线器，可禁止电路板上的硬件以使低功耗模式下的电流减小。演示板上包括晶振、RC 或固定振荡器模式，用于连接 9 伏电源适配器或电池的 5 伏稳压器， DB-9 RS-232 接口，用于通过 ICSP 进行编程和利用 MPLAB ICD 2 进行开发的 ICD 连接器，2 x 16 液晶显示器，用于 H 桥电机驱动器的 PCB 引脚布局， LIN 收发器及 EEPROM。该演示板还具备：扩展头、8 个 LED、4 个电位器、 3 个按钮开关以及实验布线区。工具包内还提供 PIC16F627A 和 PIC18F1320 样片各一枚。教程固件以及用户指南也含在演示板工具包中。

21.20 PICDEM 17 演示板

PICDEM 17 演示板是一种评估板，可以演示多种 Microchip 单片机的功能，包括 PIC17C752、PIC17C756A、PIC17C762 和 PIC17C766。它包含了一枚已编程的样片。用户可使用 PRO MATE II 器件编程器或 PICSTART Plus 开发编程器根据自己的应用开发对器件进行再编程。PICDEM 17 演示板支持从外部电路板的闪存存储器下载或执行程序。板上还配置有宽大的实验布线区供用户扩展硬件。

21.21 PICDEM 18R PIC18C601/801 演示板

PICDEM 18R 演示板用于协助用户进行 Microchip PIC18C601/801 系列单片机的开发。它用硬件实现了 8 位多路信号复用 / 信号分离和 16 位存储器模式。这块板包含 2 Mb 外部闪存存储器、128 Kb SRAM 存储器以及串行 EEPROM，允许访问 PIC18C601/801 支持的各种存储器类型。

21.22 PICDEM LIN PIC16C43X 演示板

功能强大的 LIN 软硬件工具包包括一系列电路板和 3 枚 PICmicro 单片机。外形小巧的 PIC16C432 和 PIC16C433 用作 LIN 通信中的从机，具备板上 LIN 收发器。PIC16F874 闪存单片机作为主机。所有这三枚单片机均经过固件编程以实现 LIN 总线通信。

21.23 PICkit™ 1 闪存入门工具包

作为一套完善的开发系统，PICkit 闪存入门工具包包含一块由多个部分组成的使用方便的电路板，可用于 8/14 引脚闪存 PIC® 单片机的编程、评估以及开发。电路板通过 USB 供电，可在简单的 Windows GUI 下工作。 PICkit 1 入门工具包中包括用户指南（在光盘上）、PICkit 1 教程软件和各种应用程序代码。该工具包还包括 MPLAB® IDE（集成开发环境）软件、软件和硬件 《8 引脚闪存 PIC ū
テ 厦藕偷阕印沸〔嶙雍鸵桓鹈 SB 接口线缆。它支持目前所有的 8/14 引脚闪存 PIC 单片机，以及许多计划中将要推出的器件。

21.24 PICDEM USB PIC16C7X5 演示板

PICDEM USB 演示板展示了 PIC16C745 和 PIC16C765 USB 单片机的功能。该板为将来的 USB 产品打下了基础。

21.25 评估和编程工具

除了 PICDEM 系列电路之外，Microchip 还为这些产品提供了一系列评估工具包和演示软件。

- 用于 Microchip 的 HCS 数据安全产品的 KEELOQ 评估和编程工具
- 用于汽车网络应用的 CAN 开发工具包
- 模拟电路设计板和滤波器设计软件
- PowerSmart 电池充电评估 / 校准工具包
- IrDA® 开发工具包
- microID 开发和 rfLab™ 开发软件
- 用于存储器评估和耐久性估算的 SEEVAL® 设计工具包
- 用于开关模式电源供电、高功率 IR 驱动器、Σ–Δ ADC 和流速传感器的 PICDEM MSC 演示板

有关演示和评估工具包的完整列表，请查阅 Microchip 公司网页以及最新的产品选型指南。

注：

22.0 电气特性

绝对最大额定值 (†)

偏置电压下的环境温度 .... -55°C 至 +125°C
储存温度 .... -65°C 至 +150°C
任一引脚 (除了 V_{DD}、 $\overline{MCLR}$ 和 RA4)相对于 Vss 的电压 .... -0.3V 至 (V_{DD}+0.3V)
V_{DD} 相对于 V_{SS} 的电压 .... -0.3V 至 +7.5V
$\overline{MCLR}$ 引脚相对于 Vss 的电压 (注 2) .... 0V 至 +13.25V
RA4 引脚相对于 Vss 的电压 .... 0V 至 +8.5V
总功耗 (注 1) 1.0W
Vss 引脚的最大输出电流 .... 300 mA
Vdd 引脚的最大输入电流 .... 250 mA
输入箝位电流 I_{IK} (V_I < 0 或 V_I > V_{DD}) .... ±20 mA
输出箝位电流 I_{OK} (V_O < 0 或 V_O > V_{DD}) .... ±20 mA
任一 I/O 引脚的最大灌电流 .... 25 mA
任一 I/O 引脚的最大拉电流 .... 25 mA
PORTA、 PORTB 和 PORTE 的最大灌电流总和 (注 3) .... 200 mA
PORTA、 PORTB 和 PORTE 的最大拉电流总和 (注 3) .... 200 mA
PORTC 和 PORTD 的最大灌电流总和 (注 3) .... 200 mA
PORTC 和 PORTD 的最大拉电流总和 (注 3) .... 200 mA

注 **1**:下面是功耗的计算公式:
$Pdis = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD}-V_{OH}) \times I_{OH}\} + \sum (V_{OL} \times I_{OL})$

2:如果 $\overline{MCLR}$/V_{PP} 引脚上的尖峰电压低于 V_{SS},感应电流大于 80mA,可能引起锁死。因此,如果要在 $\overline{MCLR}$/V_{PP} 引脚施加 "低电平",应串联一个 50-100Ω 的电阻,而不是将引脚直接拉到 V_{SS}。

3:PIC18F2X2 器件没有 PORTD 和 PORTE。

† 注:如果器件运行条件超过上述 "绝对最大额定值",可能会对器件造成永久性损坏。上述值仅仅是运行条件的极限值,并不意味着器件要在上述条件下或者超出上述规范范围的条件下运行。器件长时间在最大额定值条件下进行操作,会影响其可靠性。

图 **22-1**： **PIC18FXX2** 电压－频率关系曲线 （工业级）

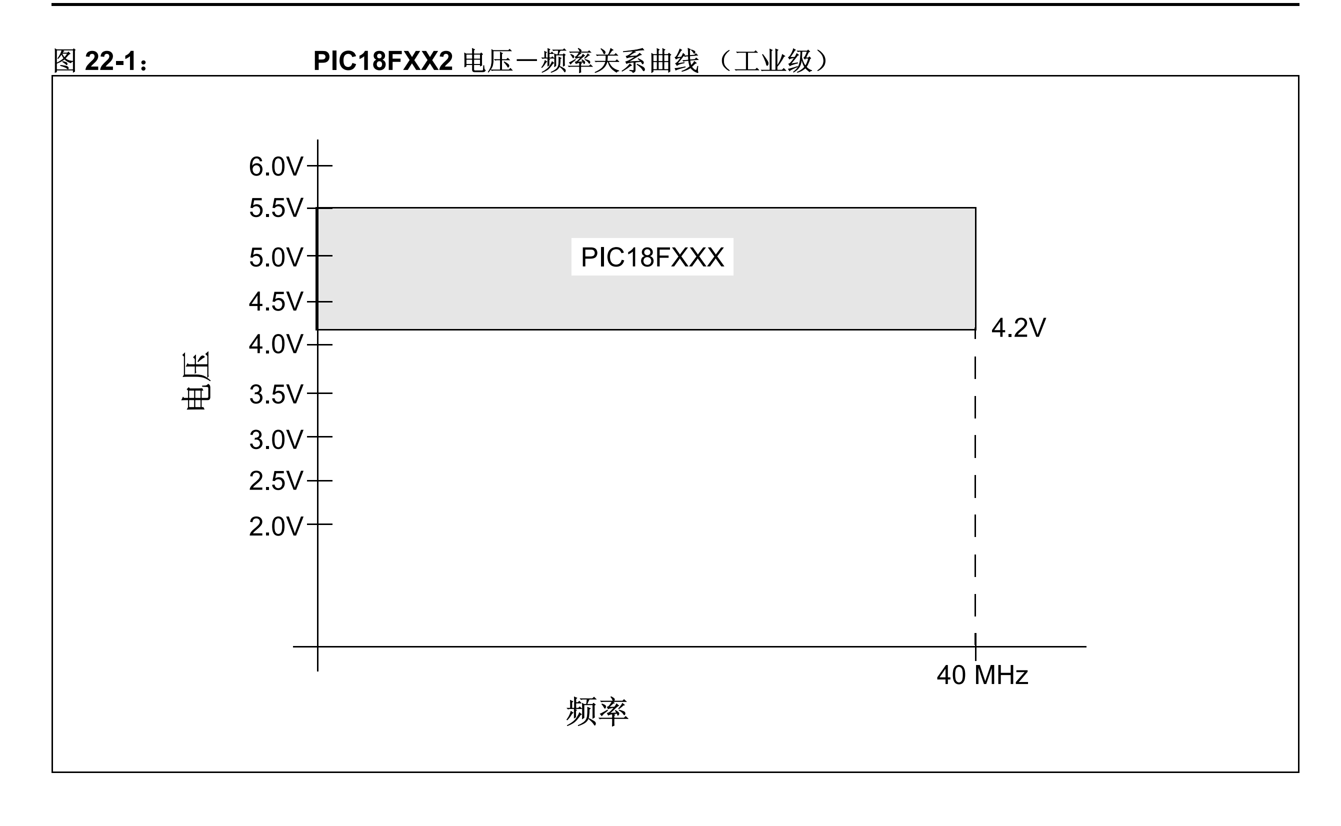
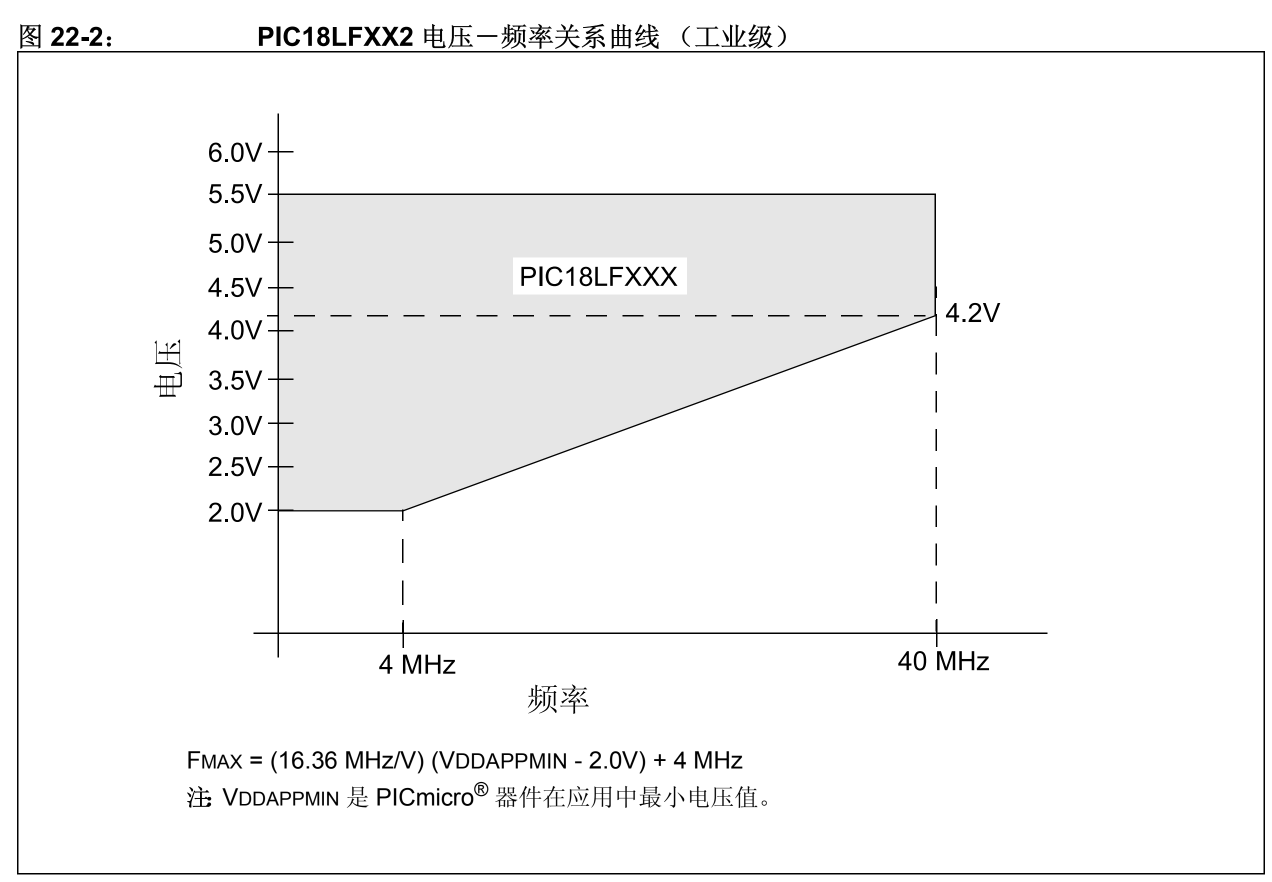


图 **22-2**： **PIC18LFXX2** 电压－频率关系曲线 （工业级）



FMAX = (16.36 MHz/V) (VDDAPPMIN - 2.0V) + 4 MHz

注: VDDAPPMIN 是 PICmicro® 器件在应用中最小电压值。

22.1 直流特性： PIC18FXX2 （工业级，扩展级） PIC18LFXX2 （工业级）

PIC18LFXX2 （工业级）			**标准运行条件 （除非另有声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
PIC18FXX2 （工业级，扩展级）			**标准运行条件 （除非另有声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数号	符号	特性	最小值	典型值	最大值	单位	条件
	VDD	电源电压					
D001		PIC18LFXX2	2.0	—	5.5	V	HS、 XT、 RC 和 LP 振荡器模式
D001		PIC18FXX2	4.2	—	5.5	V	
D002	VDR	**RAM 数据保存电压** [(1)]	1.5	—	—	V	
D003	VPOR	**VDD 启动电压** 确保内部上电复位信号	—	—	0.7	V	详情请见第 3.1 节 （上电复位）
D004	SVDD	**VDD 上升率** 确保内部上电复位信号	0.05	—	—	V/ms	详情请见第 3.1 节 （上电复位）
	VBOR	欠压复位电压					
D005		PIC18LFXX2					
		BORV1:BORV0 = 11	1.98	—	2.14	V	85°C ≥ T ≥ 25°C
		BORV1:BORV0 = 10	2.67	—	2.89	V	
		BORV1:BORV0 = 01	4.16	—	4.5	V	
		BORV1:BORV0 = 00	4.45	—	4.83	V	
D005		PIC18FXX2					
		BORV1:BORV0 = 1x	N.A.	—	N.A.	V	不在器件的工作电压范围内
		BORV1:BORV0 = 01	4.16	—	4.5	V	
		BORV1:BORV0 = 00	4.45	—	4.83	V	

图注： 阴影行有助于增强表的可读性。

注 1： 这是休眠模式下或器件复位过程中，在不丢失 RAM 数据的情况下 VDD 的最低值。

2： 供电电流主要随运行电压和频率而变化。其他因素，如 I/O 引脚负载和开关频率、振荡器类型、内部代码执行模式和温度，也会影响电流消耗。
在有源运行模式下，所有 IDD 测量值的测试条件为：
OSC1= 外部方波，满幅；所有 I/O 引脚均为三态，拉至 VDD。
MCLR=VDD ； WDT 按规定使能 / 禁止。

3： 休眠模式下的掉电电流与振荡器类型无关。掉电电流是在以下情况下测得的：器件处于休眠状态，所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，以及禁止了所有会增大△电流的功能部件（诸如 WDT、 Timer1 振荡器和 BOR 等）。

4： 对于 RC 振荡器配置，不包括流经 REXT 的电流。流经该电阻的电流可以由以下公式估算 Ir=VDD/2REXT(mA)，其中 REXT 的单位为 kΩ。

5： LVD 和 BOR 模块共用大部分电路。 ΔIBOR 和 ΔILVD 电流不会叠加。一旦使能其中一个模块，另一模块也被使能，但不会有更多损耗。

22.1 直流特性： PIC18FXX2 （工业级，扩展级） PIC18LFXX2 （工业级）（续）

PIC18LFXX2 （工业级）			标准运行条件 （除非另有声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
PIC18FXX2 （工业级，扩展级）			标准运行条件 （除非另有声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数号	符号	特性	最小值	典型值	最大值	单位	条件
	IDD	供电电流 (2,4)					
D010		PIC18LFXX2					XT 振荡器配置
			—	.5	1	mA	VDD=2.0V， +25°C， FOSC=4 MHz
			—	.5	1.25	mA	VDD=2.0V， -40°C 至 +85°C， FOSC=4 MHz
			—	1.2	2	mA	VDD=4.2V， -40°C 至 +85°C， FOSC=4 MHz
							RC 振荡器配置
			—	.3	1	mA	VDD=2.0V， +25°C， FOSC=4 MHz
			—	.3	1	mA	VDD=2.0V， -40°C 至 +85°C， FOSC=4 MHz
			—	1.5	3	mA	VDD=4.2V， -40°C 至 +85°C， FOSC=4 MHz
							RCIO 振荡器配置
			—	.3	1	mA	VDD=2.0V， +25°C， FOSC=4 MHz
			—	.3	1	mA	VDD=2.0V， -40°C 至 +85°C， FOSC=4 MHz
			—	.75	3	mA	VDD=4.2V， -40°C 至 +85°C， FOSC=4 MHz
D010		PIC18FXX2					XT 振荡器配置
			—	1.2	1.5	mA	VDD=4.2V， +25°C， FOSC=4 MHz
			—	1.2	2	mA	VDD=4.2V， -40°C 至 +85°C， FOSC=4 MHz
			—	1.2	3	mA	VDD=4.2V， -40°C 至 +125°C， FOSC=4 MHz
							RC 振荡器配置
			—	1.5	3	mA	VDD=4.2V， +25°C， FOSC=4 MHz
			—	1.5	4	mA	VDD=4.2V， -40°C 至 +85°C， FOSC=4 MHz
			—	1.6	4	mA	VDD=4.2V， -40°C 至 +125°C， FOSC=4 MHz
							RCIO 振荡器配置
			—	.75	2	mA	VDD=4.2V， +25°C， FOSC=4 MHz
			—	.75	3	mA	VDD=4.2V， -40°C 至 +85°C， FOSC=4 MHz
			—	.8	3	mA	VDD=4.2V， -40°C 至 +125°C， FOSC=4 MHz
D010A		PIC18LFXX2					LP 振荡器， FOSC=32 kHz，禁止 WDT
			—	14	30	µA	VDD=2.0V， -40°C 至 +85°C
D010A		PIC18FXX2					LP 振荡器， FOSC=32 kHz， WDT 禁止
			—	40	70	µA	VDD=4.2V， -40°C 至 +85°C
			—	50	100	µA	VDD=4.2V， -40°C 至 +125°C

图注： 阴影行有助于增强表的可读性。

注 1： 这是休眠模式下或器件复位过程中，在不丢失 RAM 数据的情况下 VDD 的最低值。

2： 供电电流主要随运行电压和频率而变化。其他因素，如 I/O 引脚负载和开关频率、振荡器类型、内部代码执行模式和温度，也会影响电流消耗。
在有源运行模式下，所有 IDD 测量值的测试条件为：
OSC1= 外部方波，满幅；所有 I/O 引脚均为三态，拉至 VDD。
MCLR=VDD ； WDT 按规定使能 / 禁止。

3： 休眠模式下的掉电电流与振荡器类型无关。掉电电流是在以下情况下测得的：器件处于休眠状态，所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，以及禁止了所有会增大△电流的功能部件（诸如 WDT、Timer1 振荡器和 BOR 等）。

4： 对于 RC 振荡器配置，不包括流经 REXT 的电流。流经该电阻的电流可以由以下公式估算 Ir=VDD/2REXT(mA)，其中 REXT 的单位为 kΩ。

5： LVD 和 BOR 模块共用大部分电路。 ΔIBOR 和 ΔILVD 电流不会叠加。一旦使能其中一个模块，另一模块也被使能，但不会有更多损耗。

22.1 直流特性： PIC18FXX2 （工业级，扩展级） PIC18LFXX2 （工业级）（续）

PIC18LFXX2 （工业级）			标准运行条件 （除非另有声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
PIC18FXX2 （工业级，扩展级）			标准运行条件 （除非另有声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数号	符号	特性	最小值	典型值	最大值	单位	条件
	IDD	供电电流 (2,4) （续）					
D010C		PIC18LFXX2	—	10	25	mA	EC 和 ECIO 振荡器配置 VDD=4.2V， -40°C 至 +85°C
D010C		PIC18FXX2	—	10	25	mA	EC 和 ECIO 振荡器配置 VDD=4.2V， -40°C 至 +125°C
D013		PIC18LFXX2	— — —	.6 10 15	2 15 25	mA mA mA	HS 振荡器配置 FOSC=4 MHz， VDD=2.0V FOSC=25 MHz， VDD=5.5V HS+PLL 振荡器配置 FOSC=10 MHz， VDD=5.5V
D013		PIC18FXX2	— —	10 15	15 25	mA mA	HS 振荡器配置 FOSC=25 MHz， VDD=5.5V HS+PLL 振荡器配置 FOSC=10 MHz， VDD=5.5V
D014		PIC18LFXX2	—	15	55	μA	Timer1 振荡器配置 FOSC=32 kHz， VDD=2.0V
D014		PIC18FXX2	— —	— —	200 250	μA μA	Timer1 振荡器配置 FOSC=32 kHz， VDD=4.2V， -40°C 至 +85°C FOSC=32 kHz， VDD=4.2V， -40°C 至 +125°C
	IPD	掉电电流 (3)					
D020		PIC18LFXX2	— — —	.08 .1 3	.9 4 10	μA μA μA	VDD=2.0V， +25°C VDD=2.0V， -40°C 至 +85°C VDD=4.2V， -40°C 至 +85°C
D020 D021B		PIC18FXX2	— — —	.1 3 15	.9 10 25	μA μA μA	VDD=4.2V， +25°C VDD=4.2V， -40°C 至 +85°C VDD=4.2V， -40°C 至 +125°C

图注： 阴影行有助于增强表的可读性。

注 1： 这是休眠模式下或器件复位过程中，在不丢失 RAM 数据的情况下 VDD 的最低值。

2： 供电电流主要随运行电压和频率而变化。其他因素，如 I/O 引脚负载和开关频率、振荡器类型、内部代码执行模式和温度，也会影响电流消耗。
在有源运行模式下，所有 IDD 测量值的测试条件为：
OSC1= 外部方波，满幅；所有 I/O 引脚均为三态，拉至 VDD。
MCLR=VDD ； WDT 按规定使能 / 禁止。

3： 休眠模式下的掉电电流与振荡器类型无关。掉电电流是在以下情况下测得的：器件处于休眠状态，所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，以及禁止了所有会增大△电流的功能部件（诸如 WDT、Timer1 振荡器和 BOR 等）。

4： 对于 RC 振荡器配置，不包括流经 REXT 的电流。流经该电阻的电流可以由以下公式估算 Ir=VDD/2REXT(mA)，其中 REXT 的单位为 kΩ。

5： LVD 和 BOR 模块共用大部分电路。 ΔIBOR 和 ΔILVD 电流不会叠加。一旦使能其中一个模块，另一模块也被使能，但不会有更多损耗。

22.1 直流特性： PIC18FXX2 （工业级，扩展级） PIC18LFXX2 （工业级）（续）

PIC18LFXX2 （工业级）			标准运行条件 （除非另有声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级）				
PIC18FXX2 （工业级，扩展级）			标准运行条件 （除非另有声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数号	符号	特性	最小值	典型值	最大值	单位	条件
		模块差分电流					
D022	ΔIWDT	看门狗定时器 PIC18LFXX2	— — —	.75 2 10	1.5 8 25	μA μA μA	VDD=2.0V， +25°C VDD=2.0V， -40°C 至 +85°C VDD=4.2V， -40°C 至 +85°C
D022		看门狗定时器 PIC18FXX2	— — —	7 10 25	15 25 40	μA μA μA	VDD=4.2V， +25°C VDD=4.2V， -40°C 至 +85°C VDD=4.2V， -40°C 至 +125°C
D022A	ΔIBOR	欠压复位 (5) PIC18LFXX2	— — —	29 29 33	35 45 50	μA μA μA	VDD=2.0V， +25°C VDD=2.0V， -40°C 至 +85°C VDD=4.2V， -40°C 至 +85°C
D022A		欠压复位 (5) PIC18FXX2	— — —	36 36 36	40 50 65	μA μA μA	VDD=4.2V， +25°C VDD=4.2V， -40°C 至 +85°C VDD=4.2V， -40°C 至 +125°C
D022B	ΔILVD	低压检测 (5) PIC18LFXX2	— — —	29 29 33	35 45 50	μA μA μA	VDD=2.0V， +25°C VDD=2.0V， -40°C 至 +85°C VDD=4.2V， -40°C 至 +85°C
D022B		低压检测 (5) PIC18FXX2	— — —	33 33 33	40 50 65	μA μA μA	VDD=4.2V， +25°C VDD=4.2V， -40°C 至 +85°C VDD=4.2V， -40°C 至 +125°C
D025	ΔITMR1	**Timer1 振荡器** PIC18LFXX2	— — —	5.2 5.2 6.5	30 40 50	μA μA μA	VDD=2.0V， +25°C VDD=2.0V， -40°C 至 +85°C VDD=4.2V， -40°C 至 +85°C
D025		**Timer1 振荡器** PIC18FXX2	— — —	6.5 6.5 6.5	40 50 65	μA μA μA	VDD=4.2V， +25°C VDD=4.2V， -40°C 至 +85°C VDD=4.2V， -40°C 至 +125°C

图注： 阴影行有助于增强表的可读性。

注 1： 这是休眠模式下或器件复位过程中，在不丢失 RAM 数据的情况下 VDD 的最低值。

2： 供电电流主要随运行电压和频率而变化。其他因素，如 I/O 引脚负载和开关频率、振荡器类型、内部代码执行模式和温度，也会影响电流消耗。
在有源运行模式下，所有 IDD 测量值的测试条件为：
OSC1= 外部方波，满幅；所有 I/O 引脚均为三态，拉至 VDD。
MCLR=VDD ； WDT 按规定使能 / 禁止。

3： 休眠模式下的掉电电流与振荡器类型无关。掉电电流是在以下情况下测得的：器件处于休眠状态，所有 I/O 引脚处于高阻态并且连接到 VDD 或 VSS，以及禁止了所有会增大△电流的功能部件（诸如 WDT、Timer1 振荡器和 BOR 等）。

4： 对于 RC 振荡器配置，不包括流经 REXT 的电流。流经该电阻的电流可以由以下公式估算 Ir=VDD/2REXT(mA)，其中 REXT 的单位为 kΩ。

5： LVD 和 BOR 模块共用大部分电路。 ΔIBOR 和 ΔILVD 电流不会叠加。一旦使能其中一个模块，另一模块也被使能，但不会有更多损耗。

22.2 直流特性： PIC18FXX2 （工业级，扩展级） PIC18LFXX2 （工业级）

直流特性			标准运行条件 （除非另有声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）			
参数号	符号	特性	最小值	最大值	单位	条件
	VIL	输入低电压				
		I/O 端口：				
D030		带 TTL 缓冲器	Vss	0.15 VDD	V	VDD ＜ 4.5V
D030A			—	0.8	V	4.5V ≤ Vdd ≤ 5.5V
D031		带施密特触发器缓冲器 RC3 和 RC4	Vss Vss	0.2 VDD 0.3 VDD	V V	
D032		$\overline{MCLR}$	VSS	0.2 VDD	V	
D032A		OSC1 （处于 XT、 HS 和 LP 模式）和 T1OSI	VSS	0.3 VDD	V	
D033		OSC1 （处于 RC 和 EC 模式）[1]	VSS	0.2 VDD	V	
	VIH	输入高电压				
		I/O 端口：				
D040		带 TTL 缓冲器	0.25 VDD + 0.8V	VDD	V	VDD ＜ 4.5V
D040A			2.0	VDD	V	4.5V ≤ Vdd ≤ 5.5V
D041		带施密特触发器缓冲器 RC3 和 RC4	0.8 VDD 0.7 VDD	VDD VDD	V V	
D042		$\overline{MCLR}$， OSC1 （EC 模式）	0.8 VDD	VDD	V	
D042A		OSC1 （处于 XT、 HS 和 LP 模式）和 T1OSI	0.7 VDD	VDD	V	
D043		OSC1 （RC 模式）[1]	0.9 VDD	VDD	V	
	IIL	输入泄漏电流 [2, 3]				
D060		I/O 端口	.02	±1	μA	VSS ≤ VPIN ≤ VDD，高阻态引脚
D061		$\overline{MCLR}$	—	±1	μA	Vss ≤ VPIN ≤ VDD
D063		OSC1	—	±1	μA	Vss ≤ VPIN ≤ VDD
	IPU	弱上拉电流				
D070	IPURB	PORTB 弱上拉电流	50	450	μA	VDD = 5V, VPIN = VSS

注 1：在 RC 振荡器配置中， OSC1/CLKI 引脚为施密特触发器输入。当处于 RC 模式时，建议不要使用外部时钟驱动 PICmicro 器件。

2：$\overline{MCLR}$ 引脚上的泄漏电流主要取决于施加的电平。规定的电平表示正常运行条件下的测量结果，在不同的输入电压下可能会测得更高的泄漏电流。

3：负电流定义为引脚的源电流。

4：该参数值仅为特征值，未经测试。

22.2 直流特性： PIC18FXX2 （工业级，扩展级） PIC18LFXX2 （工业级）（续）

直流特性			标准运行条件 （除非另有声明） 工作温度 -40°C ≤ T_A ≤ +85°C （工业级） -40°C ≤ T_A ≤ +125°C （扩展级）			
参数号	符号	特性	最小值	最大值	单位	条件
	V_{OL}	输出低电压				
D080		I/O 端口：	—	0.6	V	I_{OL} = 8.5 mA，V_{DD} = 4.5V，-40°C 至 +85°C
D080A			—	0.6	V	I_{OL} = 7.0 mA，V_{DD} = 4.5V，-40°C 至 +125°C
D083		OSC2/CLKO （RC 模式）	—	0.6	V	I_{OL}=1.6 mA，V_{DD}=4.5V，-40°C 至 +85°C
D083A			—	0.6	V	I_{OL}=1.2 mA， V_{DD} =4.5V，-40°C 至 +125°C
	V_{OH}	输出高电压 (3)				
D090		I/O 端口	V_{DD} - 0.7	—	V	I_{OH}=-3.0 mA， V_{DD}=4.5V，-40°C 至 +85°C
D090A			V_{DD} - 0.7	—	V	I_{OH}=-2.5 mA， V_{DD}=4.5V，-40°C 至 +125°C
D092		OSC2/CLKO （RC 模式）	V_{DD} - 0.7	—	V	I_{OH}=-1.3 mA,V_{DD}=4.5V, -40°C 至 +85°C
D092A			V_{DD} - 0.7	—	V	I_{OH}=-1.0 mA,V_{DD}=4.5V, -40°C 至 +125°C
D150	V_{OD}	开漏高电压	—	8.5	V	RA4 引脚
		输出引脚上的容性负载规范				
D100(4)	C_{OSC2}	OSC2 引脚	—	15	pF	XT、 HS 和 LP 模式下，使用外部时钟来驱动 OSC1 时
D101	C_{IO}	所有 I/O 引脚和 OSC2 （RC 模式）	—	50	pF	满足 AC 定时规范
D102	C_B	SCL， SDA	—	400	pF	处于 I²C 模式

注 1：在 RC 振荡器配置中， OSC1/CLKI 引脚为施密特触发器输入。当处于 RC 模式时，建议不要使用外部时钟驱动 PICmicro 器件。
2：MCLR 引脚上的泄漏电流主要取决于施加的电平。规定的电平表示正常运行条件下的测量结果，在不同的输入电压下可能会测得更高的泄漏电流。
3：负电流定义为引脚的源电流。
4：该参数值仅为特征值，未经测试。

图 **22-3**：　　低压检测特性

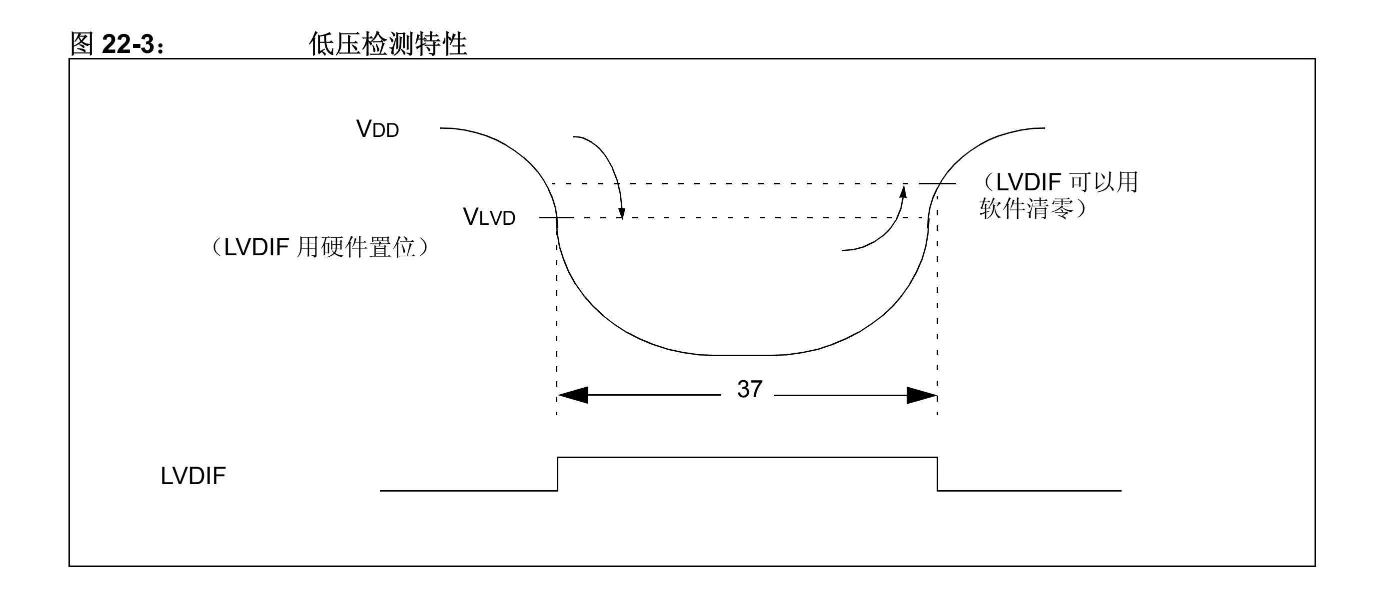


表 **22-1**：　　低压检测特性

				标准运行条件 （除非另有声明） 工作温度　-40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数号	符号	特性		最小值	典型值	最大值	单位	条件
D420	V_{LVD}	V_{DD} 上的 LVD 电压由高到低转换	LVV = `0001`	1.98	2.06	2.14	V	T ≥ 25°C
			LVV = `0010`	2.18	2.27	2.36	V	T ≥ 25°C
			LVV = `0011`	2.37	2.47	2.57	V	T ≥ 25°C
			LVV = `0100`	2.48	2.58	2.68	V	
			LVV = `0101`	2.67	2.78	2.89	V	
			LVV = `0110`	2.77	2.89	3.01	V	
			LVV = `0111`	2.98	3.1	3.22	V	
			LVV = `1000`	3.27	3.41	3.55	V	
			LVV = `1001`	3.47	3.61	3.75	V	
			LVV = `1010`	3.57	3.72	3.87	V	
			LVV = `1011`	3.76	3.92	4.08	V	
			LVV = `1100`	3.96	4.13	4.3	V	
			LVV = `1101`	4.16	4.33	4.5	V	
			LVV = `1110`	4.45	4.64	4.83	V	

表 **22-2**： 存储器编程要求

直流特性			标准运行条件 （除非另有声明） 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级）				
参数号	符号	特性	最小值	典型值 †	最大值	单位	条件
		内部程序存储器编程规范					
D110	VPP	$\overline{MCLR}$/VPP 引脚上的电压	9.00	—	13.25	V	
D113	IDDP	编程期间的源电流	—	—	10	mA	
		数据 EEPROM 存储器					
D120	ED	电池耐久性	100K	1M	—	E/W	-40°C 至 +85°C
D121	VDRW	读 / 写时的 VDD	VMIN	—	5.5	V	使用 EECON 来读 / 写 VMIN = 最小工作电压
D122	TDEW	擦 / 写周期时间	—	4	—	ms	
D123	TRETD	特性保持期	40	—	—	年	假如没有违反其他规范
D124	TREF	刷新之前，擦 / 写周期总数 (1)	1M	10M	—	E/W	-40°C 至 +85°C
		程序闪存存储器					
D130	EP	电池耐久性	10K	100K	—	E/W	-40°C 至 +85°C
D131	VPR	读取时的 VDD	VMIN	—	5.5	V	VMIN = 最小工作电压
D132	VIE	块擦除时的 VDD	4.5	—	5.5	V	使用 ICSP 端口
D132A	VIW	外部定时擦写时的 VDD	4.5	—	5.5	V	使用 ICSP 端口
D132B	VPEW	自定时写时的 VDD	VMIN	—	5.5	V	VMIN = 最小工作电压
D133	TIE	ICSP 块擦除周期时间	—	4	—	ms	VDD ≥ 4.5V
D133A	TIW	ICSP 擦写周期时间 （外部定时）	1	—	—	ms	VDD ≥ 4.5V
D133A	TIW	自定时写周期时间	—	2	—	ms	
D134	TRETD	特性保持期	40	—	—	年	假如没有违反其他规范

† 除非另有声明，“典型值”一栏中的数据均在 5.0V， 25°C 下测得。这些参数仅供设计参考，未经测试。

注 **1**：关于数据 EEPROM 耐久性更详细的讨论请参见第 6.8 节。

22.3 交流 （定时）特性

22.3.1 定时参数符号

定时参数符号根据以下格式之一创建：

1. TppS2ppS
2. TppS
3. TCC:ST （仅适用于 I^2C 规范）
4. Ts （仅适用于 I^2C 规范）

T			
F	频率	T	时间

小写字母 （pp）及其含意：

pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ 或 $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	数据输入	t0	T0CKI
io	I/O 端口	t1	T1CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$

大写字母及其含意：

S			
F	下降	P	周期
H	高	R	上升
I	无效 （高阻态）	V	有效
L	低	Z	高阻态
仅适用于 I^2C			
AA	输出访问	High	高
BUF	总线空闲	Low	低

TCC:ST （仅适用于 I^2C 规范）

CC			
HD	保持	SU	建立
ST			
DAT	数据输入保持	STO	停止状态
STA	启动状态		

22.3.2 定时条件

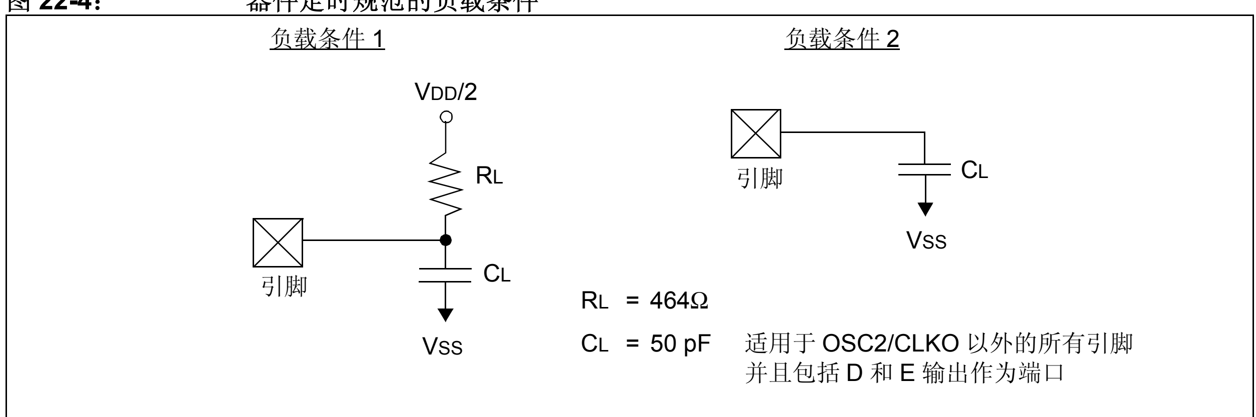
除非另外指明，表 22-3 中指定的温度和电压适用于所有定时规范。图 22-4 规定了定时规范的负载条件。

表 22-3： 温度和电压规范－交流

交流特性	**标准运行条件（除非另有声明）** 工作温度 -40°C ≤ TA ≤ +85°C （工业级） -40°C ≤ TA ≤ +125°C （扩展级） 工作电压 VDD 范围在直流规范的第 22.1 节和第 22.2 节中有描述。 LC 部件仅适合在工业温度下操作。



图 22-4： 器件定时规范的负载条件



22.3.3 时序图和规范

图 22-5： 外部时钟时序图 （除 **PLL** 以外的所有模式）

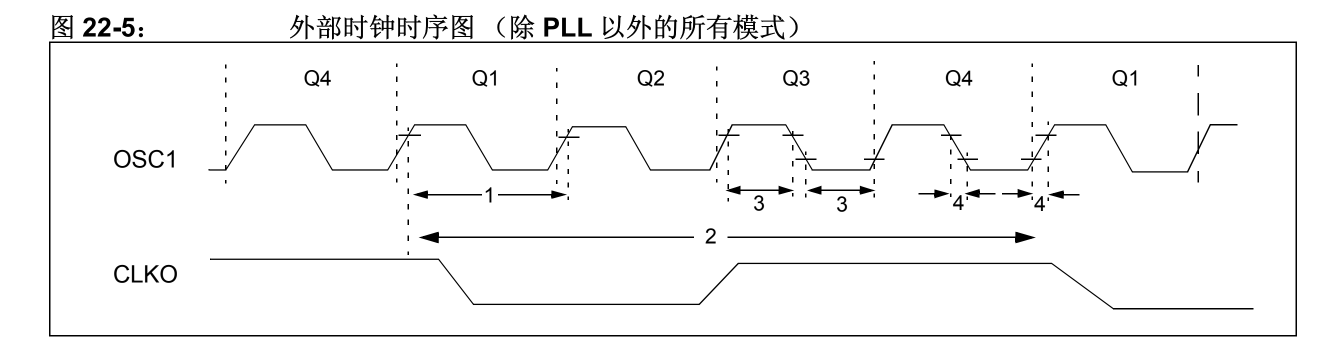


表 22-4： 外部时钟时序要求

参数号	符号	特性	最小值	最大值	单位	条件
1A	FOSC	外部 CLKI 频率 (1)	DC	40	MHz	EC， ECIO， -40°C 至 +85°C
		振荡器频率 (1)	DC	25	MHz	EC， ECIO， +85°C 至 +125°C
			DC	4	MHz	RC 振荡器
			0.1	4	MHz	XT 振荡器
			4	25	MHz	HS 振荡器
			4	10	MHz	HS+PLL 振荡器，-40°C 至 +85°C
			4	6.25	MHz	HS+PLL 振荡器，+85°C 至 +125°C
			5	200	kHz	LP 振荡器
1	TOSC	外部 CLKI 周期 (1)	25	—	ns	EC， ECIO， -40°C 至 +85°C
		振荡器周期 (1)	40	—	ns	EC， ECIO， +85°C 至 +125°C
			250	—	ns	RC 振荡器
			250	10,000	ns	XT 振荡器
			40	250	ns	HS 振荡器
			100	250	ns	HS+PLL 振荡器，-40°C 至 +85°C
			160	250	ns	HS+PLL 振荡器，+85°C 至 +125°C
			25	—	μs	LP 振荡器
2	TCY	指令周期时间 (1)	100	—	ns	TCY=4/FOSC， -40°C 至 +85°C
			160	—	ns	TCY=4/FOSC， +85°C 至 +125°C
3	TosL，TosH	外部时钟输入 （OSC1）高电平或低电平时间	30	—	ns	XT 振荡器
			2.5	—	μs	LP 振荡器
			10	—	ns	HS 振荡器
4	TosR，TosF	外部时钟输入 （OSC1）上升或下降时间	—	20	ns	XT 振荡器
			—	50	ns	LP 振荡器
			—	7.5	ns	HS 振荡器

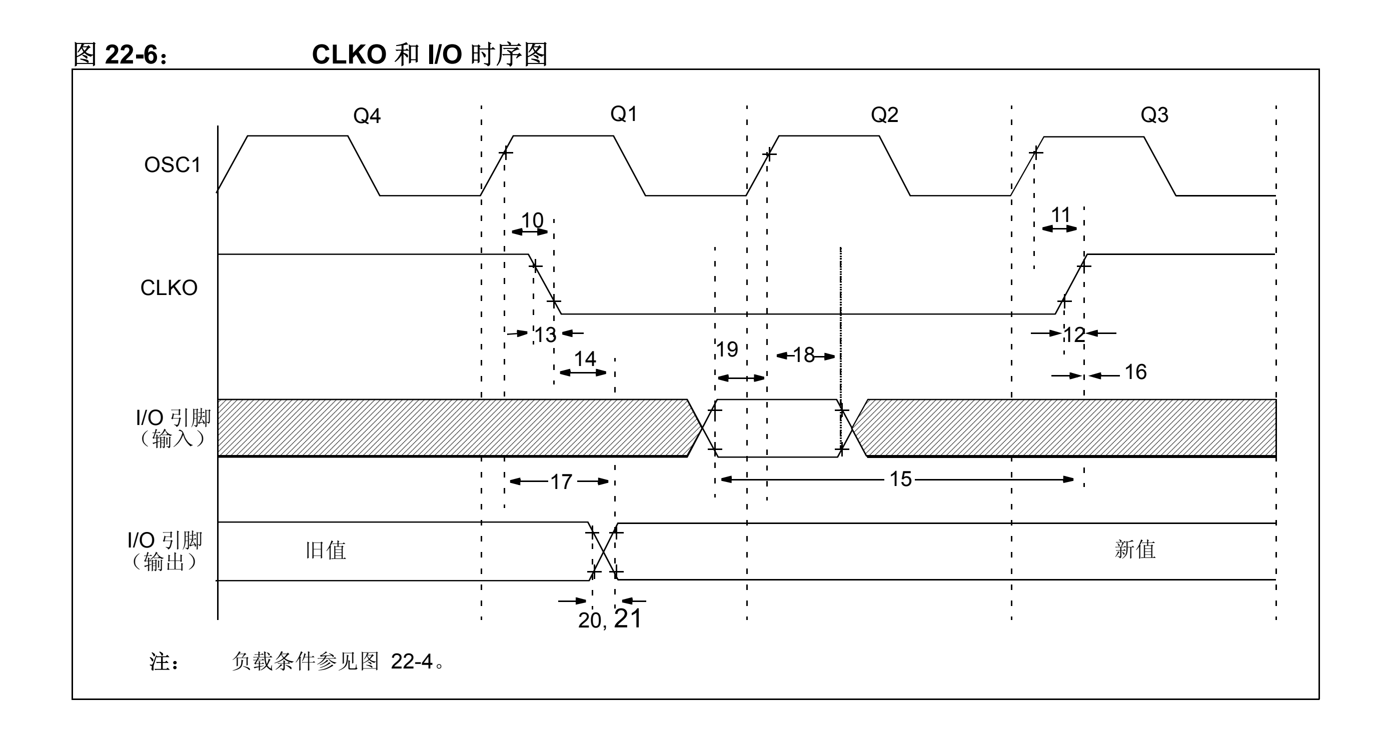
注 **1：** 对于除了 PLL 之外的所有配置，指令周期 （TCY）等于输入振荡器时基周期的四倍。所有规定值均基于器件执行代码在标准运行条件下所对应的特定振荡器类型的特征数据。超过这些规定值可能导致振荡器运行不稳定和 / 或电流消耗比预期的要高。所有器件在测试 “最小值”时，都将外部时钟接到 OSC1/CLKI 引脚。当使用了外部时钟输入时，所有器件的 “最大”周期时间极限为 “DC”（没有时钟）。

表 22-5: **PLL 时钟定时规范 (VDD=4.2 至 5.5V)**

参数号	符号	特性	最小值	典型值 †	最大值	单位	条件
—	FOSC	振荡器频率范围	4	—	10	MHz	仅 HS 模式
—	FSYS	片内 VCO 系统频率	16	—	40	MHz	仅 HS 模式
—	t_{rc}	PLL 启动时间 (锁定时间)	—	—	2	ms	
—	△ CLK	CLKO 稳定性 (抖动)	-2	—	+2	%	

† 除非另有声明,“典型值”一栏中的数据均在 5.0V, 25°C 下测得。这些参数仅供设计参考,未经测试。

图 22-6: **CLKO 和 I/O 时序图**



注: 负载条件参见图 22-4。

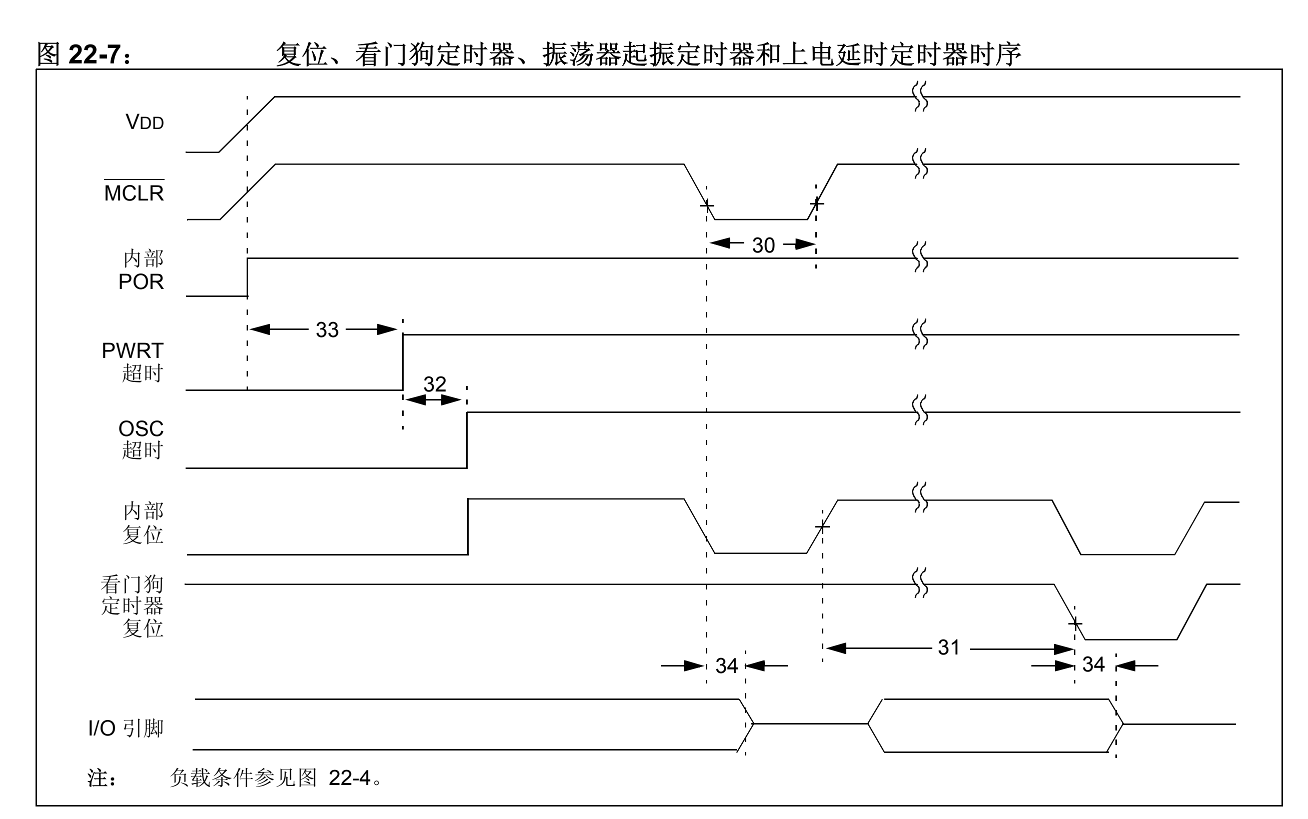
表 22-6： CLKO 和 I/O 时序要求

参数号	符号	特性		最小值	典型值	最大值	单位	条件
10	TosH2ckL	OSC1↑ 到 CLKO↓		—	75	200	ns	(注 1)
11	TosH2ckH	OSC1↑ 到 CLKO↑		—	75	200	ns	(注 1)
12	TckR	CLKO 上升时间		—	35	100	ns	(注 1)
13	TckF	CLKO 下降时间		—	35	100	ns	(注 1)
14	TckL2ioV	CLKO↑ 到端口输出有效		—	—	0.5 TCY+20	ns	(注 1)
15	TioV2ckH	在 CLKO↑ 前端口输入有效		0.25 TCY+25	—	—	ns	(注 1)
16	TckH2ioI	在 CLKO↑ 后端口输入保持		0	—	—	ns	(注 1)
17	TosH2ioV	OSC1↑ （Q1 周期）到端口输出有效		ó	50	150	ns	
18	TosH2ioI	OSC1↑ （Q2 周期）到端口输入无效 （I/O 在保持时间）	PIC18**F**XXX	100	—	—	ns	
18A			PIC18**LF**XXX	200	—	—	ns	
19	TioV2osH	端口输入有效到 OSC↑ （I/O 在建立时间）		0	—	—	ns	
20	TioR	端口输出上升时间	PIC18**F**XXX	—	10	25	ns	
20A			PIC18**LF**XXX	—	—	60	ns	VDD = 2V
21	TioF	端口输出下降时间	PIC18**F**XXX	—	10	25	ns	
21A			PIC18**LF**XXX	—	—	60	ns	VDD = 2V
22††	TINP	INT 引脚高电平或低电平时间		TCY	—	—	ns	
23††	TRBP	RB7:RB4 改变 INT 高电平或低电平时间		TCY	—	—	ns	
24††	TRCP	RC7:RC4 改变 INT 高电平或低电平时间		20			ns	

†† 这些参数是与任何内部时钟边沿沿无关的异步事件。

注 1：测量在 RC 模式下进行，其中 CLKO 输出为 4 xTOSC。

图 22-7： 复位、看门狗定时器、振荡器起振定时器和上电延时定时器时序



注： 负载条件参见图 22-4。

图 22-8： 欠压复位时序图

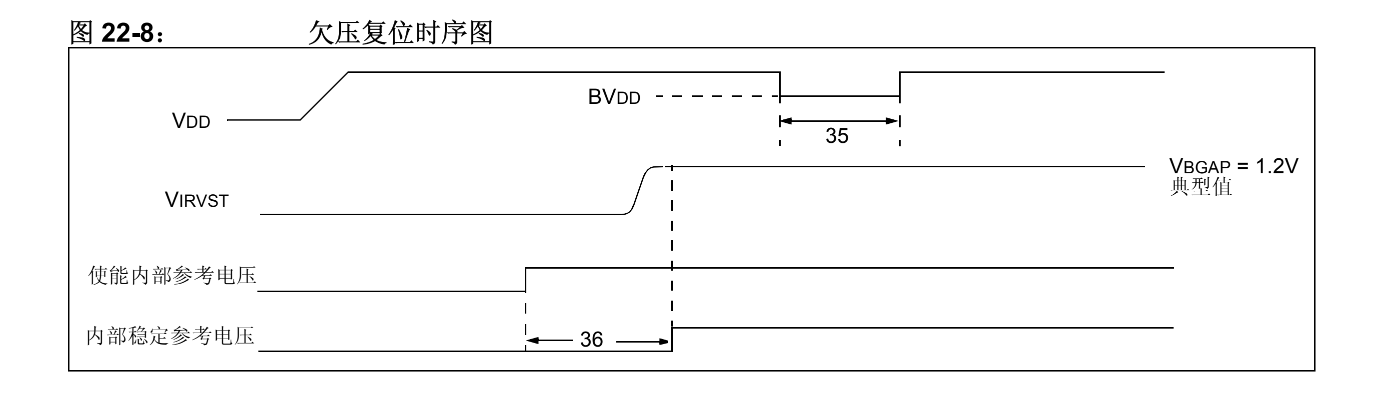


表 22-7： 复位、看门狗定时器、振荡器起振定时器、上电延时定时器和欠压复位要求

参数号	符号	特性	最小值	典型值	最大值	单位	条件
30	TmcL	$\overline{\text{MCLR}}$ 脉冲宽度 （低）	2	—	—	μs	
31	TWDT	看门狗定时器超时周期 （无后分频器）	7	18	33	ms	
32	TOST	振荡器起振定时器周期	1024 TOSC	—	1024 TOSC	—	TOSC=OSC1 周期
33	TPWRT	上电延时定时器周期	28	72	132	ms	
34	TIOZ	$\overline{\text{MCLR}}$ 低电平或看门狗定时器复位引起 I/O 高阻态	—	2	—	μs	
35	TBOR	欠压复位脉冲宽度	200	—	—	μs	VDD ≤ BVDD （见 D005）
36	TIVRST	内部参考电压稳定所需的时间	—	20	500	μs	
37	TLVD	低压检测脉冲宽度	200	—	—	μs	VDD ≤ VLVD （见 D420）

图 22-9： TIMER0 和 TIMER1 外部时钟时序图

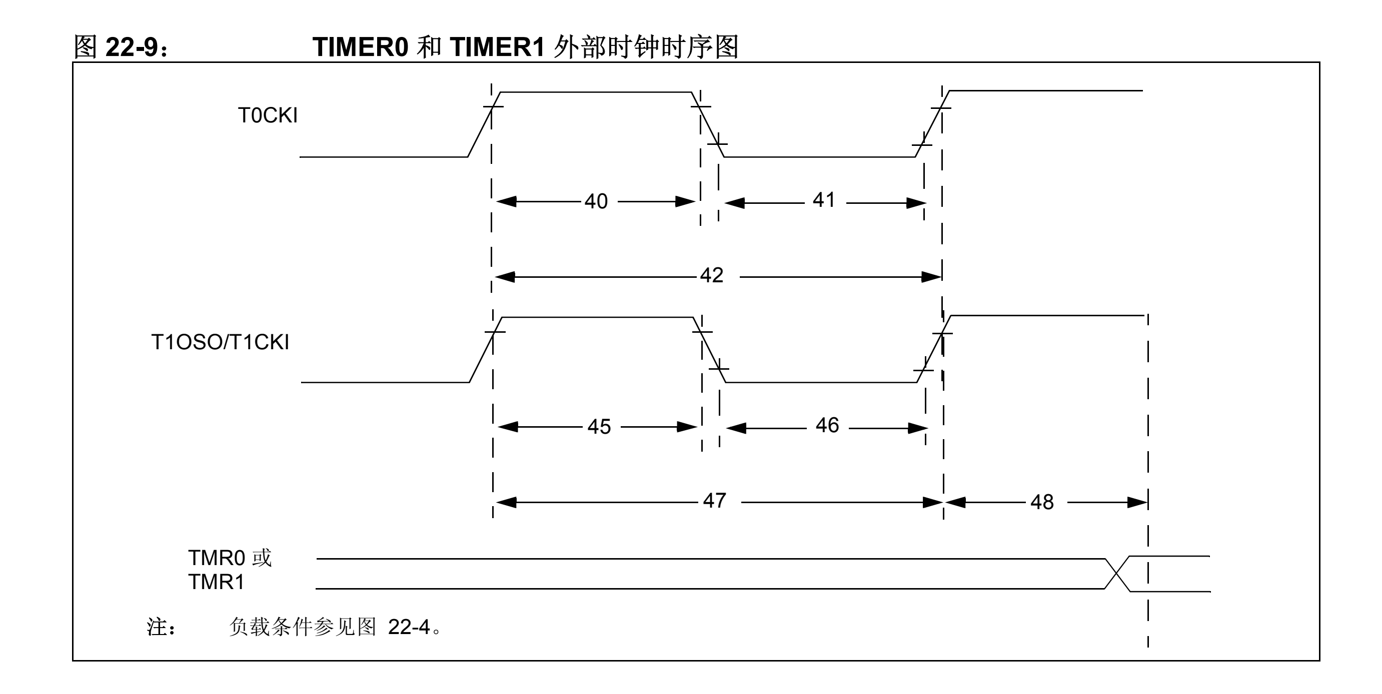
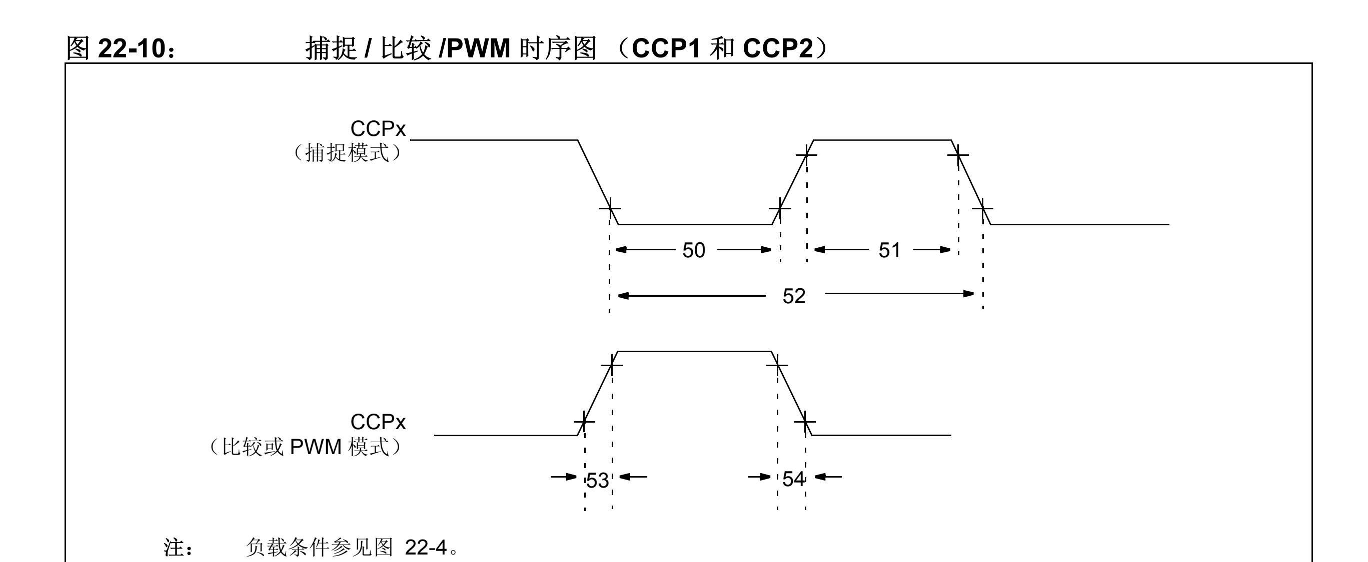


表 22-8： TIMER0 和 TIMER1 的外部时钟要求

参数号	符号	特性			最小值	最大值	单位	条件
40	Tt0H	T0CKI 高电平脉冲宽度		无预分频器	0.5TCY + 20	—	ns	
				有预分频器	10	—	ns	
41	Tt0L	T0CKI 低电平脉冲宽度		无预分频器	0.5TCY + 20	—	ns	
				有预分频器	10	—	ns	
42	Tt0P	T0CKI 周期		无预分频器	TCY +10	—	ns	
				有预分频器	取较大值： 20 ns 或 $\frac{Tcy + 40}{N}$	—	ns	N= 预分频值 (1, 2, 4,..., 256)
45	Tt1H	T1CKI 高电平时间	同步，无分频器		0.5TCY + 20	—	ns	
			同步， 有预分频器	PIC18**F**XXX	10	—	ns	
				PIC18**LF**XXX	25	—	ns	
			异步	PIC18**F**XXX	30	—	ns	
				PIC18**LF**XXX	50	—	ns	
46	Tt1L	T1CKI 低电平时间	同步，无分频器		0.5TCY +5	—	ns	
			同步， 有预分频器	PIC18**F**XXX	10	—	ns	
				PIC18**LF**XXX	25	—	ns	
			异步	PIC18**F**XXX	30	—	ns	
				PIC18**LF**XXX	50	—	ns	
47	Tt1P	T1CKI 输入周期	同步		取较大值： 20 ns 或 $\frac{Tcy + 40}{N}$	—	ns	N = 预分 频值 (1, 2, 4, 8)
			异步		60	—	ns	
	Ft1	T1CKI 振荡器输入频率范围			DC	50	kHz	
48	Tcke2tmrI	从外部 T1CKI 时钟边沿到定时器增 1 的延时			2TOSC	7TOSC	—	

图 **22-10**： 捕捉 / 比较 /PWM 时序图 （CCP1 和 CCP2）



注： 负载条件参见图 22-4。

表 **22-9**： 捕捉 / 比较 /PWM 要求 （CCP1 和 CCP2）

参数号	符号	特性			最小值	最大值	单位	条件
50	TccL	CCPx 输入低电平时间	无预分频器		0.5 TCY + 20	—	ns	
			带预分频器	PIC18**F**XXX	10	—	ns	
				PIC18**LF**XXX	20	—	ns	
51	TccH	CCPx 输入高电平时间	无预分频器		0.5 TCY + 20	—	ns	
			带预分频器	PIC18**F**XXX	10	—	ns	
				PIC18**LF**XXX	20	—	ns	
52	TccP	CCPx 输入周期			(3 TCY + 40) / N	—	ns	N= 预分频值 (1,4 或 16)
53	TccR	CCPx 输出上升时间		PIC18**F**XXX	—	25	ns	
				PIC18**LF**XXX	—	60	ns	Vdd = 2V
54	TccF	CCPx 输出下降时间		PIC18**F**XXX	—	25	ns	
				PIC18**LF**XXX	—	60	ns	Vdd = 2V

图 22-11： 并行从动端口时序图 （PIC18F4X2）

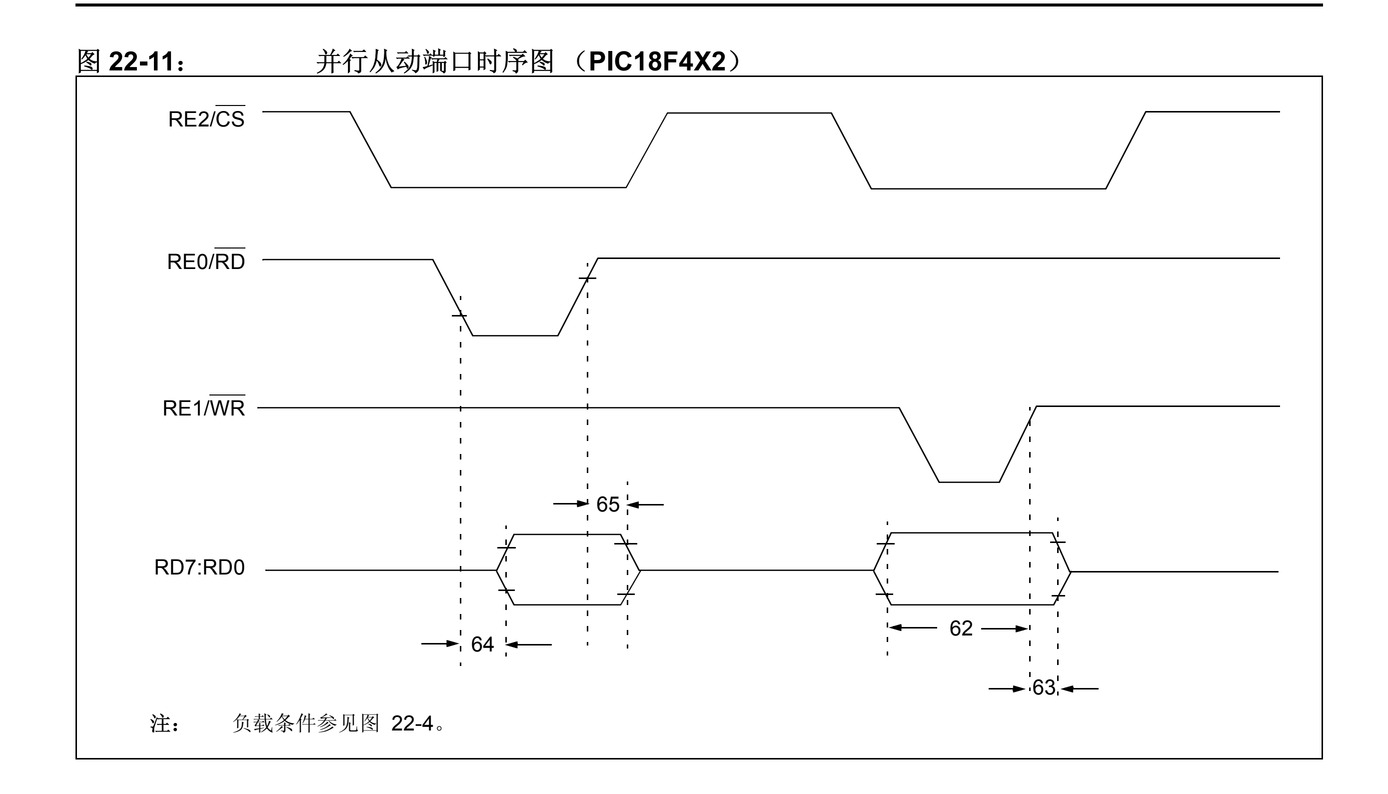
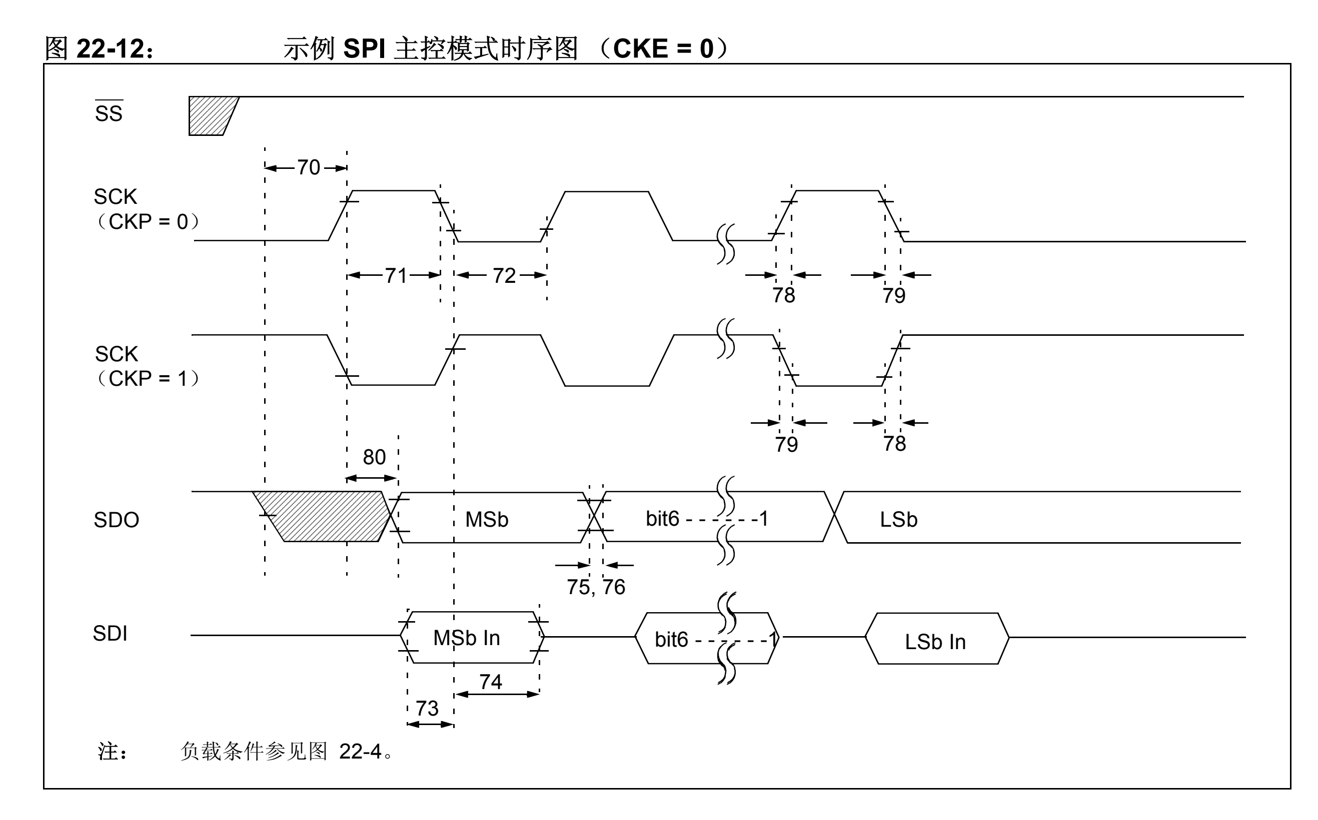


表 22-10： 并行从动端口要求 （PIC18F4X2）

参数号	符号	特性		最小值	最大值	单位	条件
62	TdtV2wrH	WR↑ 或 CS↑ 前数据输入有效时间（建立时间）		20 25	— —	ns ns	 扩展级温度范围
63	TwrH2dtI	WR↑ 或 CS↑ 到数据输入无效时间 （保持时间）	PIC18**F**XXX	20	—	ns	
			PIC18**LF**XXX	35	—	ns	Vdd = 2V
64	TrdL2dtV	RD↓ 和 CS↓ 到数据输出有效		—	80 90	ns ns	 扩展级温度范围
65	TwrH2dtI	RD↑ 或 CS↓ 到数据输出无效		10	30	ns	
66	TibfINH	禁止 IBF 标志位被 WR↑ 或 CS↑ 清零		—	3 TCY		

图 **22-12**： 示例 **SPI** 主控模式时序图 （**CKE = 0**）



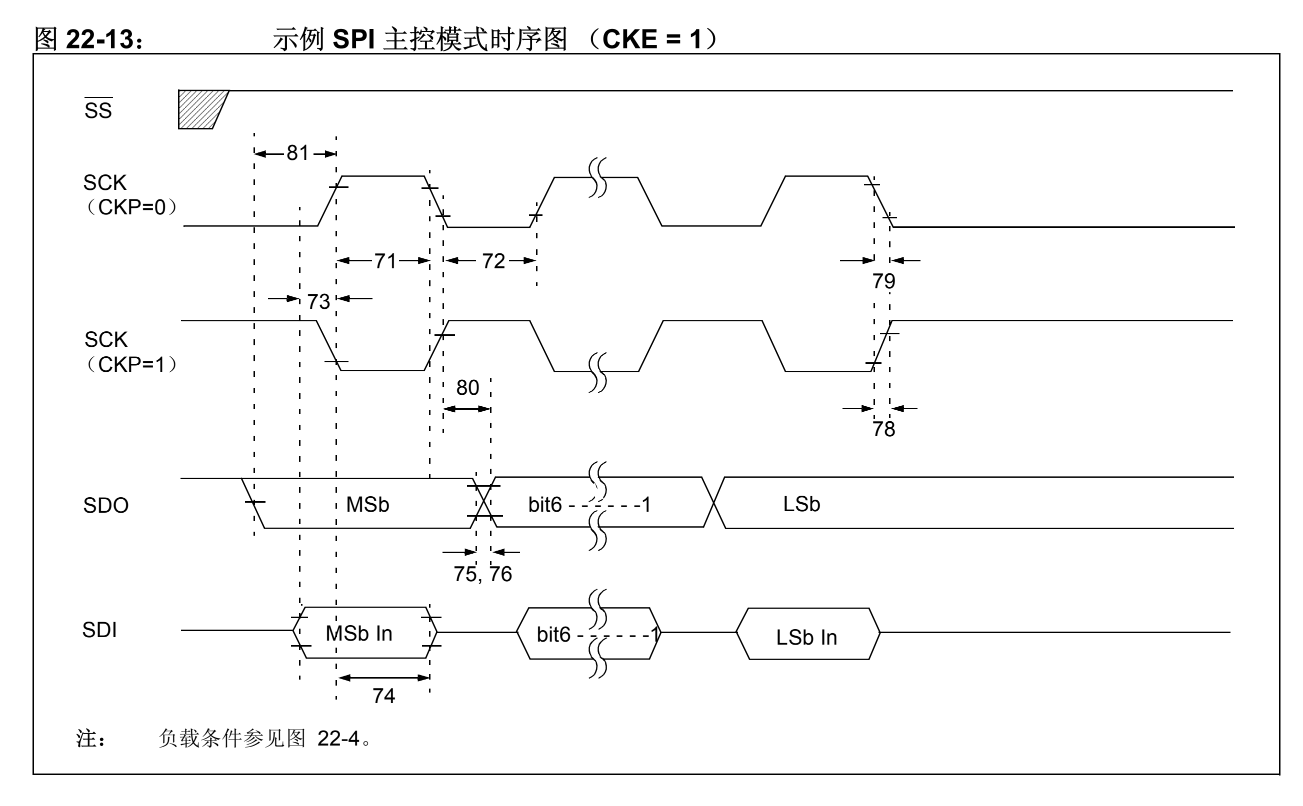
注： 负载条件参见图 22-4。

表 **22-11**： 示例 **SPI** 模式要求 （主控模式， **CKE = 0**）

参数号	符号	特性		最小值	最大值	单位	条件
70	TssL2scH, TssL2scL	$\overline{SS}$↓ 到 SCK↓ 或 SCK↑ 输入		TCY	—	ns	
71	TscH	SCK 输入高电平时间 （从动模式）	连续	1.25 TCY + 30	—	ns	
71A			单字节	40	—	ns	**(** 注 **1)**
72	TscL	SCK 输入低电平时间 （从动模式）	连续	1.25 TCY + 30	—	ns	
72A			单字节	40	—	ns	**(** 注 **1)**
73	TdiV2scH, TdiV2scL	SDI 数据输入到 SCK 边沿的建立时间		100	—	ns	
73A	TB2B	Byte1 的最后一个时钟边沿到 Byte2 的首个时钟边沿		1.5 TCY + 40	—	ns	**(** 注 **2)**
74	TscH2diL, TscL2diL	SDI 数据输入到 SCK 边沿的保持时间		100	—	ns	
75	TdoR	SDO 数据输出上升时间	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
76	TdoF	SDO 数据输出下降时间	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
78	TscR	SCK 输出上升时间 （主控模式）	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
79	TscF	SCK 输出下降时间 （主控模式）	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
80	TscH2doV, TscL2doV	SCK 边沿后的 SDO 数据输出有效	PIC18**F**XXX	—	50	ns	
			PIC18**LF**XXX	—	150	ns	Vdd = 2V

注 **1**：要求使用参数 #73A。
2：仅当使用参数 #71A 和 #72A 时。

图 22-13： 示例 SPI 主控模式时序图 （CKE = 1）



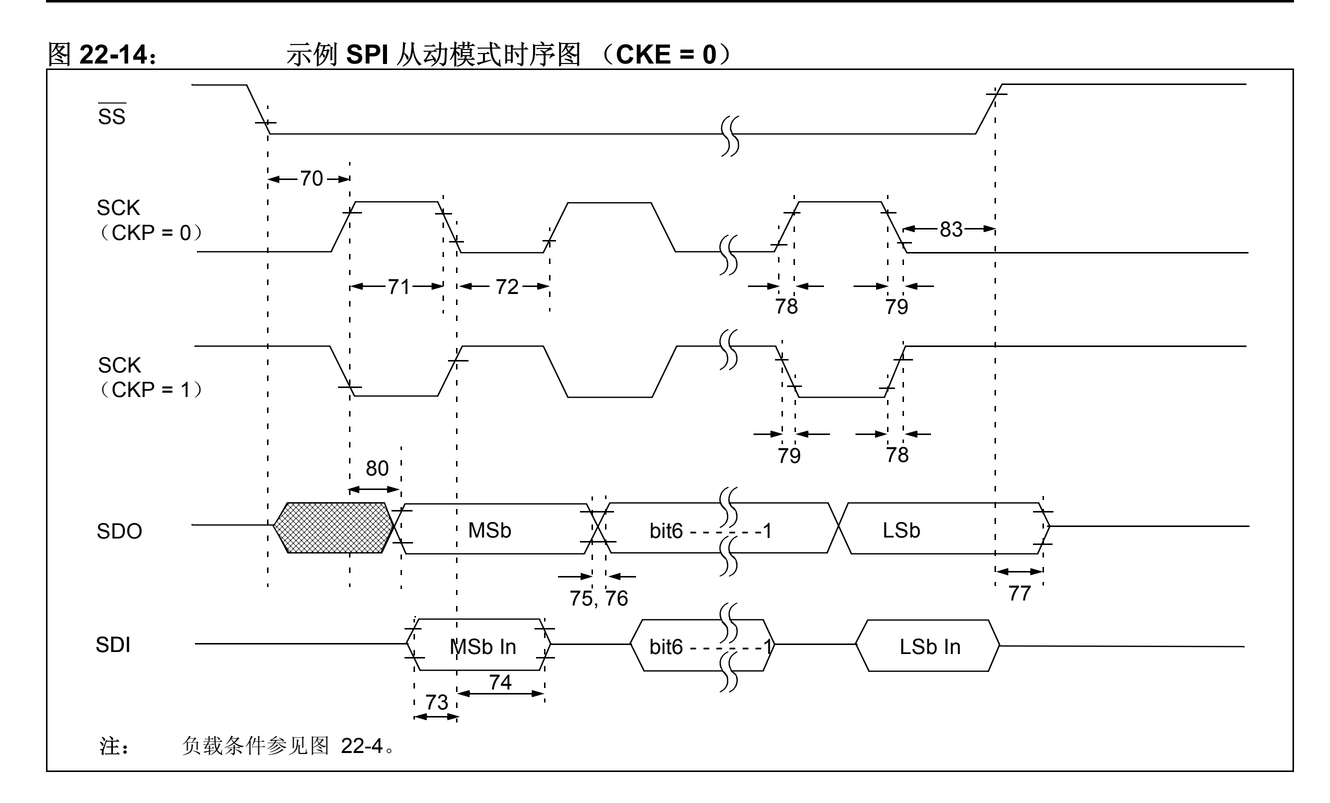
注： 负载条件参见图 22-4。

表 22-12： 示例 SPI 模式要求 （主控模式， CKE = 1）

参数号	符号	特性		最小值	最大值	单位	条件
71	TscH	SCK 输入高电平时间（从动模式）	连续	1.25 TCY + 30	—	ns	
71A			单字节	40	—	ns	(注 1)
72	TscL	SCK 输入低电平时间（从动模式）	连续	1.25 TCY + 30	—	ns	
72A			单字节	40	—	ns	(注 1)
73	TdiV2scH, TdiV2scL	SDI 数据输入到 SCK 边沿的建立时间		100	—	ns	
73A	TB2B	Byte1 的最后一个时钟边沿到 Byte2 的首个时钟边沿		1.5 TCY + 40	—	ns	(注 2)
74	TscH2diL, TscL2diL	SDI 数据输入到 SCK 边沿的保持时间		100	—	ns	
75	TdoR	SDO 数据输出上升时间	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
76	TdoF	SDO 数据输出下降时间	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
78	TscR	SCK 输出上升时间 （主控模式）	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
79	TscF	SCK 输出下降时间 （主控模式）	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
80	TscH2doV TscL2doV	SCK 边沿后 SDO 数据输出有效	PIC18**F**XXX	—	50	ns	
			PIC18**LF**XXX	—	150	ns	Vdd = 2V
81	TdoV2scH TdoV2scL	SDO 数据输出建立到 SCK 边沿		TCY	—	ns	

注 **1**：要求使用参数 #73A。
2：仅当使用参数 #71A 和 #72A 时。

图 22-14： 示例 SPI 从动模式时序图 （CKE = 0）



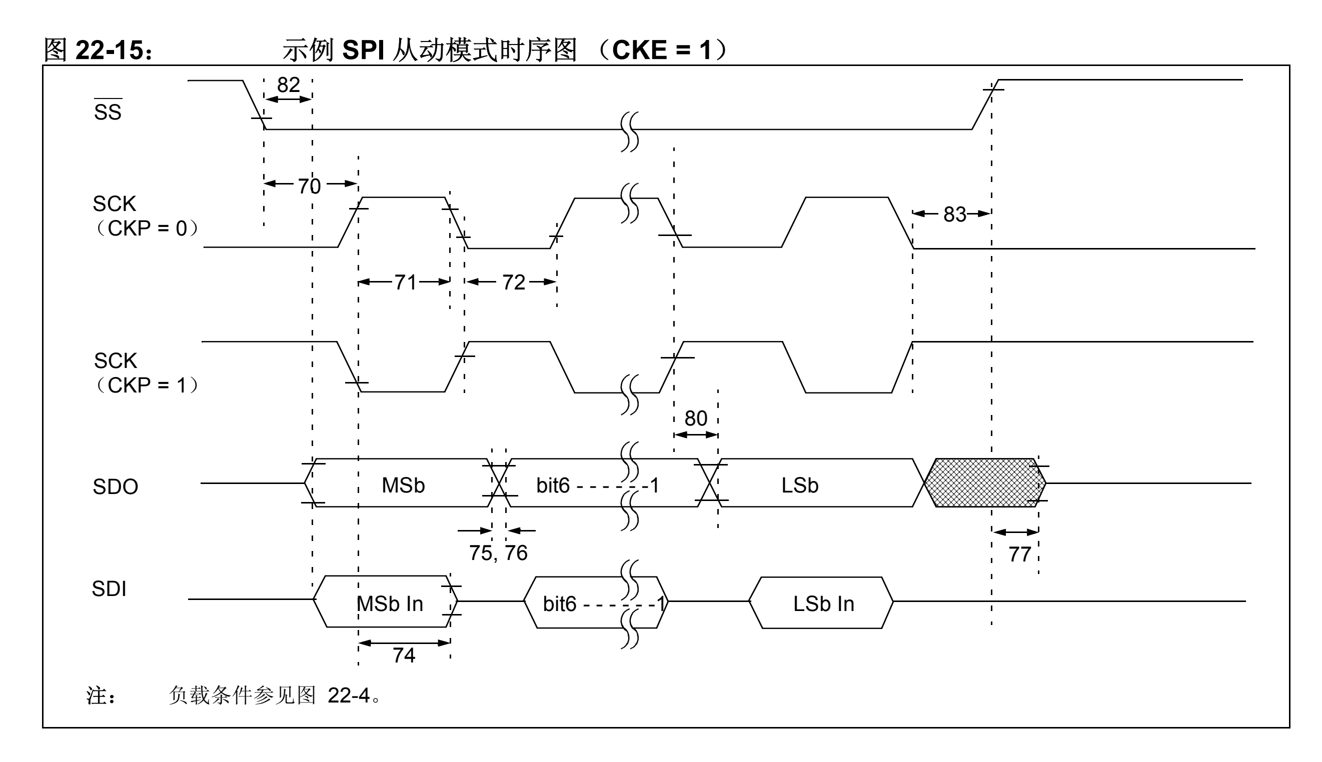
注： 负载条件参见图 22-4。

表 22-13： 示例 SPI 模式要求 （从动模式时序， CKE = 0）

参数 号	符号	特性		最小值	最大值	单位	条件
70	TssL2scH TssL2scL	$\overline{SS}$↓ 到 SCK↓ 或 SCK↑ 输入		TCY	—	ns	
71	TscH	SCK 输入高电平时间 （从动模式）	连续	1.25 TCY + 30	—	ns	
71A			单字节	40	—	ns	**(注 1)**
72	TscL	SCK 输入低电平时间 （从动模式）	连续	1.25 TCY + 30	—	ns	
72A			单字节	40	—	ns	**(注 1)**
73	TdiV2scH TdiV2scL	SDI 数据输出到 SCK 边沿的建立时间		100	—	ns	
73A	TB2B	Byte1 的最后一个时钟边沿到 Byte2 的首个时钟边沿		1.5 TCY + 40	—	ns	**(注 2)**
74	TscH2diL TscL2diL	SDI 数据输入到 SCK 边沿的保持时间		100	—	ns	
75	TdoR	SDO 数据输出上升时间	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
76	TdoF	SDO 数据输出下降时间	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
77	TssH2doZ	$\overline{SS}$↑ 到 SDO 输出高阻态		10	50	ns	
78	TscR	SCK 输出上升时间 （主控模式）	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
79	TscF	SCK 输出下降时间 （主控模式）	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
80	TscH2doV TscL2doV	SCK 边沿后 SDO 数据输出有效	PIC18**F**XXX	—	50	ns	
			PIC18**LF**XXX	—	150	ns	Vdd = 2V
83	TscH2ssH TscL2ssH	SCK 边沿后 $\overline{SS}$ ↑		1.5 Tcy + 40	—	ns	

注 **1：** 要求使用参数 #73A。
2： 仅当使用参数 #71A 和 #72A 时。

图 22-15： 示例 SPI 从动模式时序图 （CKE = 1）



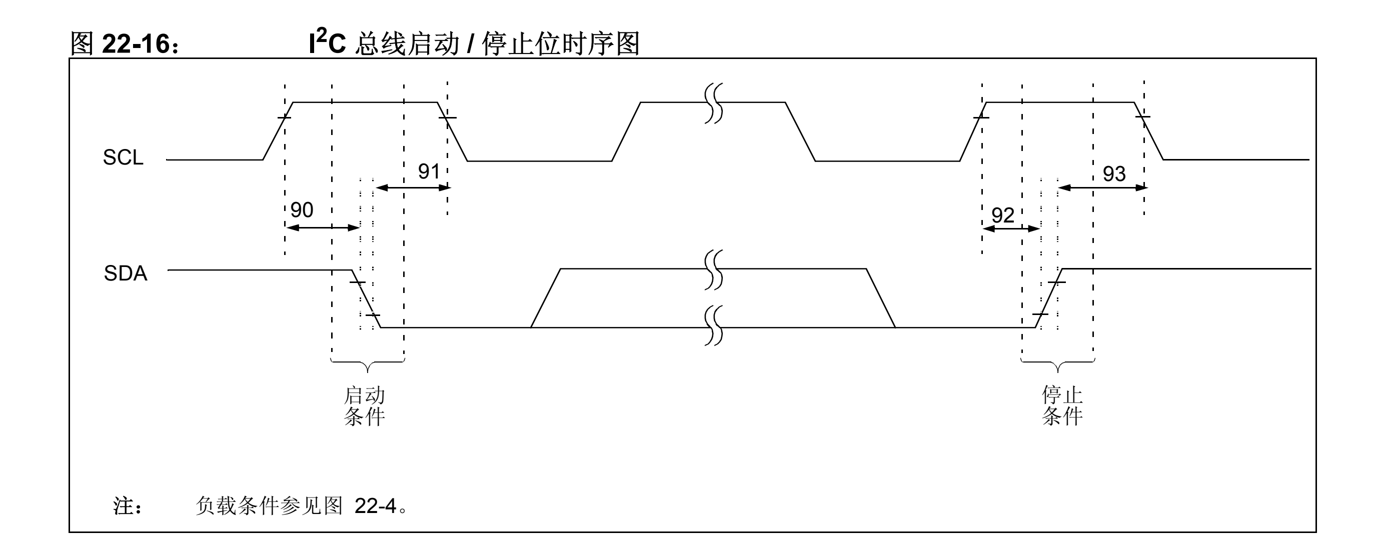
注： 负载条件参见图 22-4。

表 22-14： 示例 SPI 从动模式要求 （CKE = 1）

参数号	符号	特性		最小值	最大值	单位	条件
70	TssL2scH TssL2scL	$\overline{SS}$↓ 到 SCK↓ 或 SCK↑ 输入		TCY	—	ns	
71	TscH	SCK 输入高电平时间（从动模式）	连续	1.25 TCY + 30	—	ns	
71A			单字节	40	—	ns	(注 1)
72	TscL	SCK 输入低电平时间（从动模式）	连续	1.25 TCY + 30	—	ns	
72A			单字节	40	—	ns	(注 1)
73A	TB2B	Byte1 的最后一个时钟边沿到 Byte2 的首个时钟边沿		1.5 TCY + 40	—	ns	(注 2)
74	TscH2diL TscL2diL	SDI 数据输入到 SCK 边沿的保持时间		100	—	ns	
75	TdoR	SDO 数据输出上升时间	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
76	TdoF	SDO 数据输出下降时间	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
77	TssH2doZ	$\overline{SS}$↑ 到 SDO 输出高阻态		10	50	ns	
78	TscR	SCK 输出上升时间 （主控模式）	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
79	TscF	SCK 输出下降时间 （主控模式）	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	Vdd = 2V
80	TscH2doV TscL2doV	SCK 边沿后 SDO 数据输出有效	PIC18**F**XXX	—	50	ns	
			PIC18**LF**XXX	—	150	ns	Vdd = 2V
82	TssL2doV	$\overline{SS}$↓ 边沿后 SDO 数据输出有效	PIC18**F**XXX	—	50	ns	
			PIC18**LF**XXX	—	150	ns	Vdd = 2V
83	TscH2ssH TscL2ssH	SCK 边沿后 $\overline{SS}$↑		1.5 TCY +40	—	ns	

注 **1：** 要求使用参数 #73A。
2： 仅当使用参数 #71A 和 #72A 时。

图 22-16： I²C 总线启动 / 停止位时序图

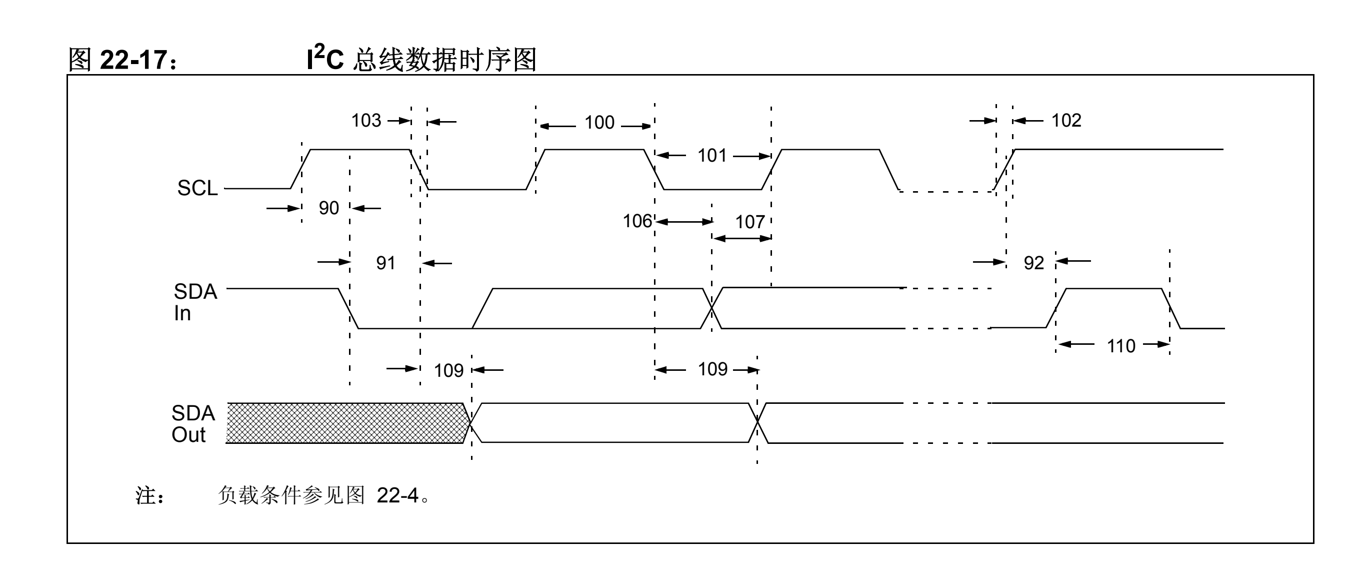


注： 负载条件参见图 22-4。

表 22-15： I²C 总线启动 / 停止位要求 （从动模式）

参数号	符号	特性		最小值	最大值	单位	条件
90	TSU:STA	启动条件	100 kHz 模式	4700	—	ns	仅与重复启动条件相关
		建立时间	400 kHz 模式	600	—		
91	THD:STA	启动条件	100 kHz 模式	4000	—	ns	这个周期后产生第一个时钟脉冲。
		保持时间	400 kHz 模式	600	—		
92	TSU:STO	停止条件	100 kHz 模式	4700	—	ns	
		建立时间	400 kHz 模式	600	—		
93	THD:STO	停止条件	100 kHz 模式	4000	—	ns	
		保持时间	400 kHz 模式	600	—		

图 22-17： I²C 总线数据时序图



注： 负载条件参见图 22-4。

表 **22-16：** **I²C** 总线数据要求 （从动模式）

参数号	符号	特性		最小值	最大值	单位	条件
100	THIGH	时钟高电平时间	100 kHz 模式	4.0	—	μs	PIC18FXXX 工作频率不得低于 1.5 MHz
			400 kHz 模式	0.6	—	μs	PIC18FXXX 工作频率不得低于 10 MHz
			SSP 模块	1.5 TCY	—		
101	TLOW	时钟低电平时间	100 kHz 模式	4.7	—	μs	PIC18FXXX 工作频率不得低于 1.5 MHz
			400 kHz 模式	1.3	—	μs	PIC18FXXX 工作频率不得低于 10 MHz
			SSP 模块	1.5 TCY	—		
102	TR	SDA 和 SCL 上升时间	100 kHz 模式	—	1000	ns	
			400 kHz 模式	20 + 0.1 CB	300	ns	规定 CB 为 10 至 400pF
103	TF	SDA 和 SCL 下降时间	100 kHz 模式	—	1000	ns	VDD ≥ 4.2V
			400 kHz 模式	20 + 0.1 CB	300	ns	VDD ≥ 4.2V
90	TSU:STA	启动状态建立时间	100 kHz 模式	4.7	—	μs	仅与重复启动条件相关
			400 kHz 模式	0.6	—	μs	
91	THD:STA	启动状态保持时间	100 kHz 模式	4.0	—	μs	这个周期后产生第一个时钟脉冲。
			400 kHz 模式	0.6	—	μs	
106	THD:DAT	数据输入保持时间	100 kHz 模式	0	—	ns	
			400 kHz 模式	0	0.9	μs	
107	TSU:DAT	数据输入建立时间	100 kHz 模式	250	—	ns	**(** 注 **2)**
			400 kHz 模式	100	—	ns	
92	TSU:STO	停止状态建立时间	100 kHz 模式	4.7	—	μs	
			400 kHz 模式	0.6	—	μs	
109	TAA	来自时钟的输出有效	100 kHz 模式	—	3500	ns	**(** 注 **1)**
			400 kHz 模式	—	—	ns	
110	TBUF	总线空闲时间	100 kHz 模式	4.7	—	μs	在开始新的发送之前，总线必须保持空闲的时间
			400 kHz 模式	1.3	—	μs	
D102	CB	总线容性负载		—	400	pF	

注 **1：** 为了避免意外产生启动和停止条件，器件作为发送方必须留出内部最小延迟时间 （最小 300 ns）经过 SCL 下降沿的未定义区域。

2： 可将快速模式 I²C 总线器件用于标准模式 I²C 总线系统，但必须满足 TSU:DAT ≥ 250 ns 的要求。如果器件不延长 SCL 信号的低电平周期，上述要求将自动满足。如果这样的一个器件延长了 SCL 信号的低电平周期，其下一个数据位必须输出到 SDA 线路。在 SCL 线路释放前， TR max.+TSU:DAT=1000+250=1250 ns （按照标准模式 I²C 总线规范）。

图 22-18： MSSP I²C 总线启动 / 停止位时序波形

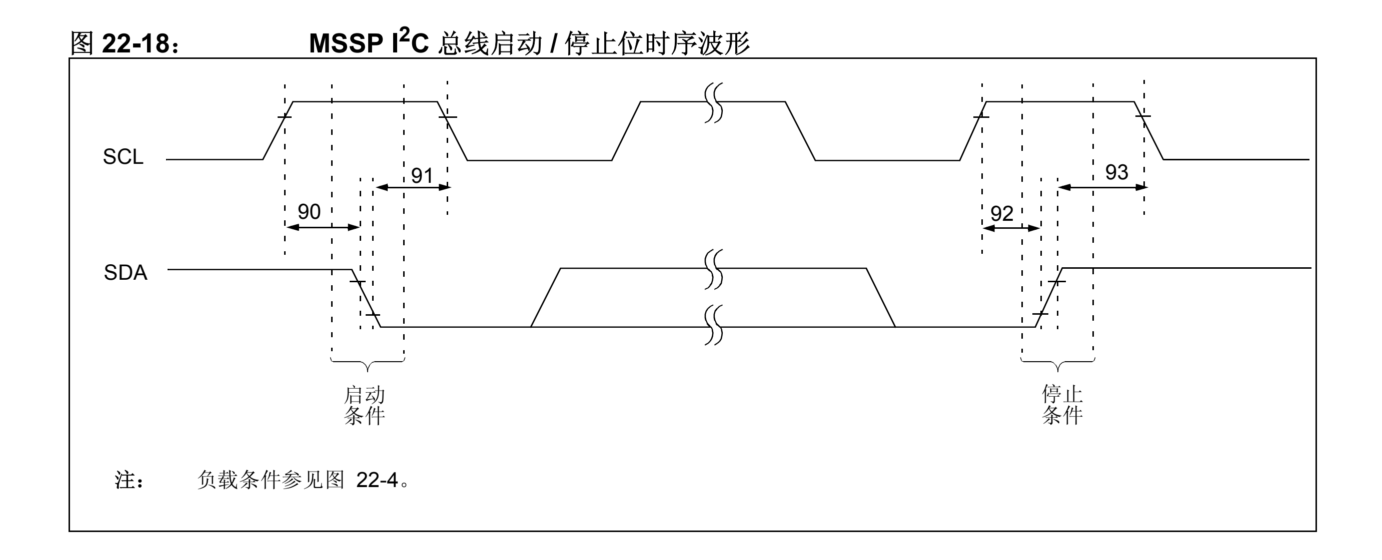


表 22-17： MSSP I²C 总线启动 / 停止位的要求

参数号	符号	特性		最小值	最大值	单位	条件
90	TSU:STA	启动条件建立时间	100 kHz 模式	2(TOSC)(BRG + 1)	—	ns	仅与重复启动条件相关
			400 kHz 模式	2(TOSC)(BRG + 1)	—		
			1 MHz 模式 (1)	2(TOSC)(BRG + 1)	—		
91	THD:STA	启动条件保持时间	100 kHz 模式	2(TOSC)(BRG + 1)	—	ns	这个周期后，产生第一个时钟脉冲
			400 kHz 模式	2(TOSC)(BRG + 1)	—		
			1 MHz 模式 (1)	2(TOSC)(BRG + 1)	—		
92	TSU:STO	停止条件建立时间	100 kHz 模式	2(TOSC)(BRG + 1)	—	ns	
			400 kHz 模式	2(TOSC)(BRG + 1)	—		
			1 MHz 模式 (1)	2(TOSC)(BRG + 1)	—		
93	THD:STO	停止条件保持时间	100 kHz 模式	2(TOSC)(BRG + 1)	—	ns	
			400 kHz 模式	2(TOSC)(BRG + 1)	—		
			1 MHz 模式 (1)	2(TOSC)(BRG + 1)	—		

注 1：对于所有 I²C 引脚，最大引脚电容 =10 pF。

图 22-19： MSSP I²C 总线数据时序图

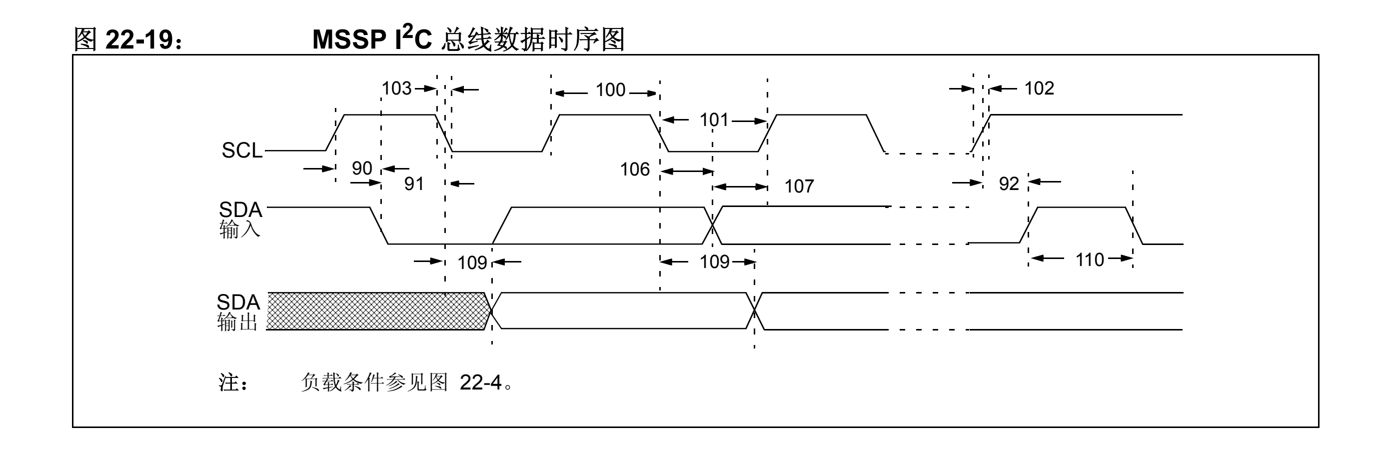


表 22-18： MSSP I²C 总线数据的要求

参数号	符号	特性		最小值	最大值	单位	条件
100	T_{HIGH}	时钟高电平时间	100 kHz 模式	2(T_{OSC})(BRG + 1)	—	ms	
			400 kHz 模式	2(T_{OSC})(BRG + 1)	—	ms	
			1 MHz 模式 (1)	2(T_{OSC})(BRG + 1)	—	ms	
101	T_{LOW}	时钟低电平时间	100 kHz 模式	2(T_{OSC})(BRG + 1)	—	ms	
			400 kHz 模式	2(T_{OSC})(BRG + 1)	—	ms	
			1 MHz 模式 (1)	2(T_{OSC})(BRG + 1)	—	ms	
102	T_R	SDA 和 SCL 上升时间	100 kHz 模式	—	1000	ns	规定 C_B 为 10 至 400pF
			400 kHz 模式	20 + 0.1 C_B	300	ns	
			1 MHz 模式 (1)	—	300	ns	
103	T_F	SDA 和 SCL 下降时间	100 kHz 模式	—	1000	ns	V_{DD} ≥ 4.2V
			400 kHz 模式	20 + 0.1 C_B	300	ns	V_{DD} ≥ 4.2V
90	$T_{SU:STA}$	启动条件建立时间	100 kHz 模式	2(T_{OSC}(BRG + 1)	—	ms	仅与重复启动条件相关
			400 kHz 模式	2(T_{OSC})(BRG + 1)	—	ms	
			1 MHz 模式 (1)	2(T_{OSC})(BRG + 1)	—	ms	
91	$T_{HD:STA}$	启动条件保持时间	100 kHz 模式	2(T_{OSC})(BRG + 1)	—	ms	这个周期后，产生第一个时钟脉冲。
			400 kHz 模式	2(T_{OSC})(BRG + 1)	—	ms	
			1 MHz 模式 (1)	2(T_{OSC})(BRG + 1)	—	ms	
106	$T_{HD:DAT}$	数据输入保持时间	100 kHz 模式	0	—	ns	
			400 kHz 模式	0	0.9	ms	
107	$T_{SU:DAT}$	数据输入建立时间	100 kHz 模式	250	—	ns	**(注 2)**
			400 kHz 模式	100	—	ns	
92	$T_{SU:STO}$	停止条件建立时间	100 kHz 模式	2(T_{OSC})(BRG + 1)	—	ms	
			400 kHz 模式	2(T_{OSC})(BRG + 1)	—	ms	
			1 MHz 模式 (1)	2(T_{OSC})(BRG + 1)	—	ms	
109	T_{AA}	来自时钟的输出有效	100 kHz 模式	—	3500	ns	
			400 kHz 模式	—	1000	ns	
			1 MHz 模式 (1)	—	—	ns	
110	T_{BUF}	总线空闲时间	100 kHz 模式	4.7	—	ms	在开始新的发送之前，总线必须保持空闲的时间
			400 kHz 模式	1.3	—	ms	
D102	C_B	总线容性负载		—	400	pF	

注 **1：** 对于所有 I²C 引脚，最大引脚电容 =10 pF。

2： 可将快速模式 I²C 总线器件用于标准模式 I²C 总线系统，但是必须满足参数 #107 ≥ 250 ns。如果器件不延长 SCL 信号的低电平周期，将自动满足该条件。如果这样的一个器件延长 SCL 信号的低电平周期，其下一个数据位必须输出到 SDA 线路。 SCL 线路释放前，参数 #102+ 参数 #107=1000+250=1250 ns （100 kHz 模式）。

图 22-20： USART 同步发送 （主控 / 从动）时序图

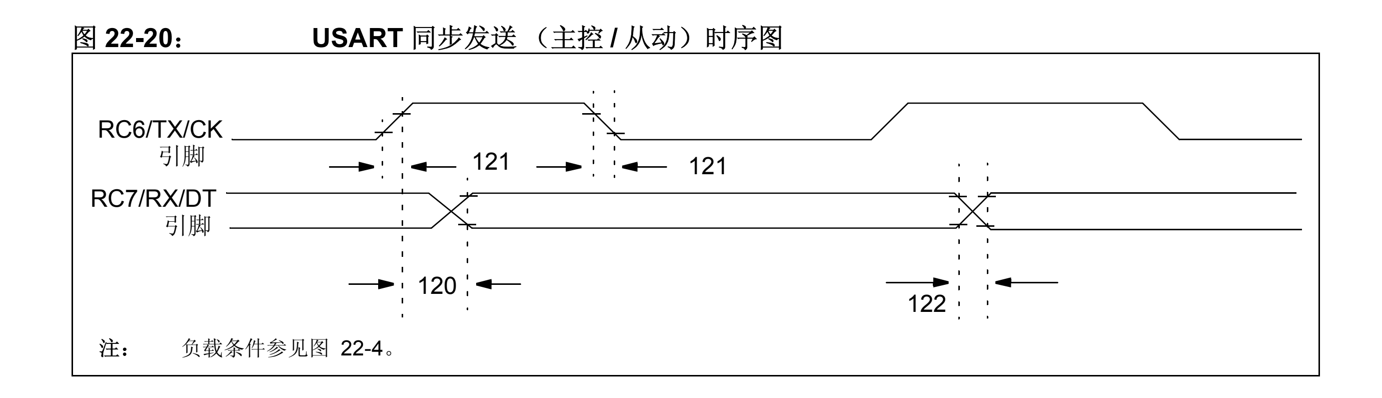


表 22-19： USART 同步发送的要求

参数号	符号	特性		最小值	最大值	单位	条件
120	TckH2dtV	同步发送 （主控和从动模式） 时钟高电平到数据输出有效	PIC18**F**XXX	—	50	ns	
			PIC18**LF**XXX	—	150	ns	VDD=2V
121	Tckr	时钟输出上升和下降时间 （主控模式）	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	VDD = 2V
122	Tdtr	数据输出上升和下降时间	PIC18**F**XXX	—	25	ns	
			PIC18**LF**XXX	—	60	ns	VDD = 2V

图 22-21： USART 同步接收 （主控 / 从动）时序图

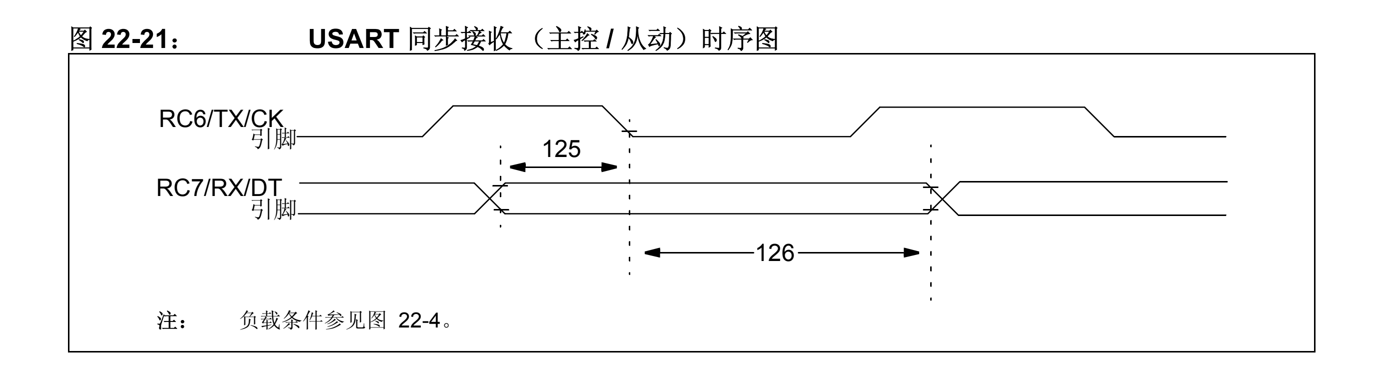


表 22-20： USART 同步接收要求

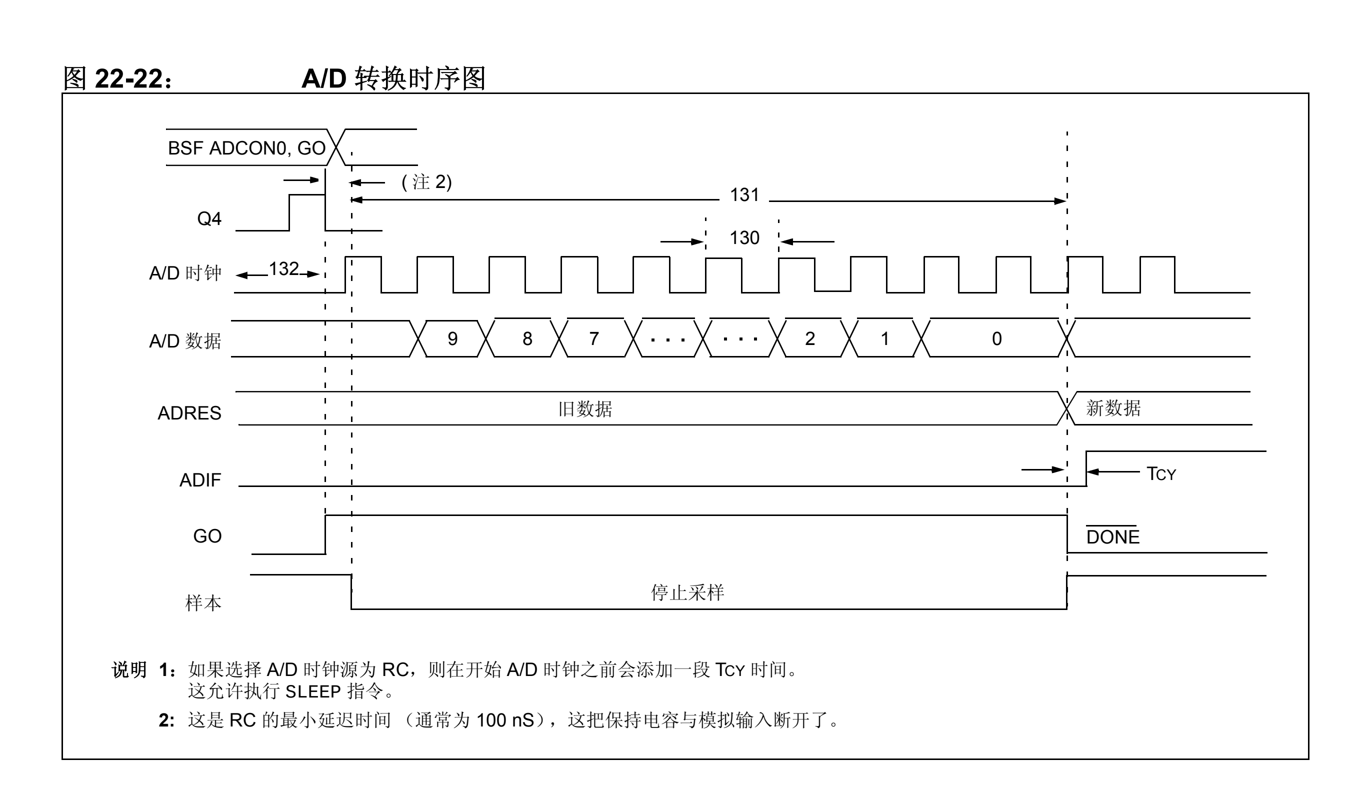
参数号	符号	特性		最小值	最大值	单位	条件
125	TdtV2ckl	同步接收 （主控和从动模式） CK↓ 前，数据保持 （DT 保持时间）		10	—	ns	
126	TckL2dtl	CK↓ 后，数据保持 （DT 保持时间）	PIC18**F**XXX	15	—	ns	
			PIC18**LF**XXX	20	—	ns	VDD = 2V

表 22-21： A/D 转换器特性： PIC18FXX2 （工业级，扩展级） PIC18LFXX2 （工业级）

参数号	符号	特性	最小值	典型值	最大值	单位	条件
A01	NR	分辨率	—	—	10	位	
A03	EIL	积分线性误差	—	—	＜ ±1	LSb	VREF = VDD = 5.0V
A04	EDL	微分线性误差	—	—	＜ ±1	LSb	VREF = VDD = 5.0V
A05	EG	增益误差	—	—	＜ ±1	LSb	VREF = VDD = 5.0V
A06	EOFF	偏移误差	—	—	＜ ±1.5	LSb	VREF = VDD = 5.0V
A10	—	单调性	保证 (2)			—	VSS ≤ VAIN ≤ VREF
A20 A20A	VREF	参考电压 （VREFH - VREFL）	1.8V 3V	— —	— —	V V	VDD ＜ 3.0V VDD ＞ 3.0V
A21	VREFH	参考电压高电平	AVSS	—	AVDD + 0.3V	V	
A22	VREFL	参考电压低电平	AVSS - 0.3V	—	VREFH	V	
A25	VAIN	模拟输入电压	AVSS - 0.3V	—	AVDD + 0.3V	V	VDD ≥ 2.5V **(注 3)**
A30	ZAIN	模拟电压源的建议阻抗	—	—	2.5	kΩ	**(注 4)**
A50	IREF	VREF 输入电流 **(注 1)**	—	— —	5 150	μA μA	在 VAIN 采集期间 在 A/D 转换周期

注 **1:** VSS ≤ VAIN ≤ VREF
2: A/D 转换结果从不随输入电压的增加而减少，并且不会丢失代码。
3: 如果 VDD ＜ 2.5V，则必须限定 VAIN ＜ 0.5 VDD。
4: 模拟电压源允许的最大阻抗为 10 kΩ。这需要更多的采集次数。

图 22-22： A/D 转换时序图



说明 1: 如果选择 A/D 时钟源为 RC，则在开始 A/D 时钟之前会添加一段 TCY 时间。这允许执行 SLEEP 指令。
2: 这是 RC 的最小延迟时间 （通常为 100 nS），这把保持电容与模拟输入断开了。

表 **22-22**： **A/D** 转换的要求

参数号	符号	特性		最小值	最大值	单位	条件
130	T_{AD}	A/D 时钟周期	PIC18**F**XXX	1.6	20[4]	μs	基于 T_{OSC}
			PIC18**F**XXX	2.0	6.0	μs	A/D RC 模式
131	T_{CNV}	转换时间 (不包括采集时间)**(注 1)**		11	12	T_{AD}	
132	T_{ACQ}	采集时间 **(注 2)**		5 10	— —	μs μs	$V_{REF} = V_{DD} = 5.0V$ $V_{REF} = V_{DD} = 2.5V$
135	T_{SWC}	转换 → 采样的切换时间		—	**(注 3)**		

注 **1**：可在后续 T_{CY} 周期内读 ADRES 寄存器。

2：当新的输入电压值与上一个采样电压值相比，变化不超过一个 LSb 时，保持电容采集这个 “新”值所需的时间。输入通道上的源阻抗 （R_S）为 50Ω。更多关于采集时间的注意事项的信息，请参见第 17.0 节。

3：器件时钟的下一个 Q4 周期。

4：A/D 时钟周期的时间取决于器件频率和 T_{AD} 时钟分频器。

23.0 DC 和 AC 特性图表

注：	以下图表为基于有限样本数的统计结果，仅供参考。这里列出的性能特性未经测试，也不作保证。某些图表中的数据超出了规定的工作范围 （例如，超出了规定的电源范围），因此超出了保证的范围。

"典型值"代表在 25°C 下的平均值，而 "最大值"代表 （平均值 + 3σ），"最小值"代表 （平均值 - 3σ），其中 σ 代表整个温度范围内的标准偏差。

图 23-1： 典型 IDD 与 FOSC 的关系曲线 （在不同 VDD 下， HS 模式）

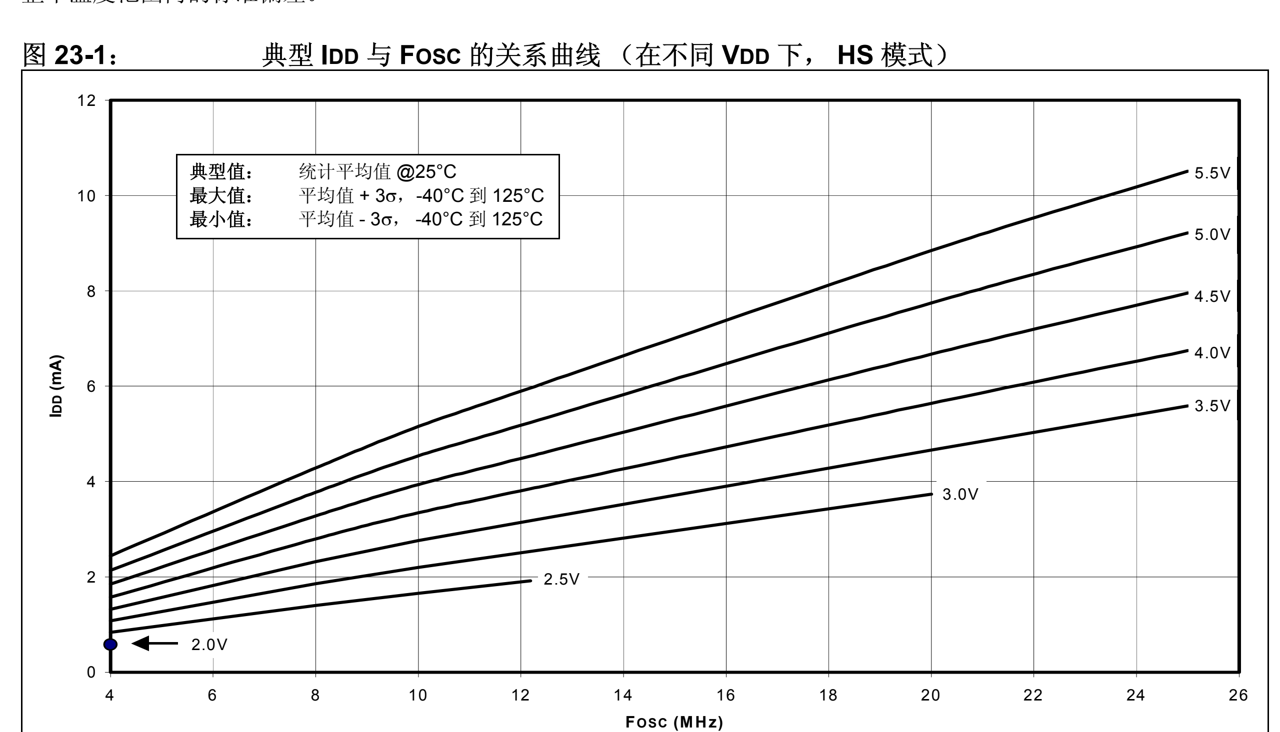


图 23-2： 最大值 IDD 与 FOSC 的关系曲线 （在不同 VDD 下， HS 模式）

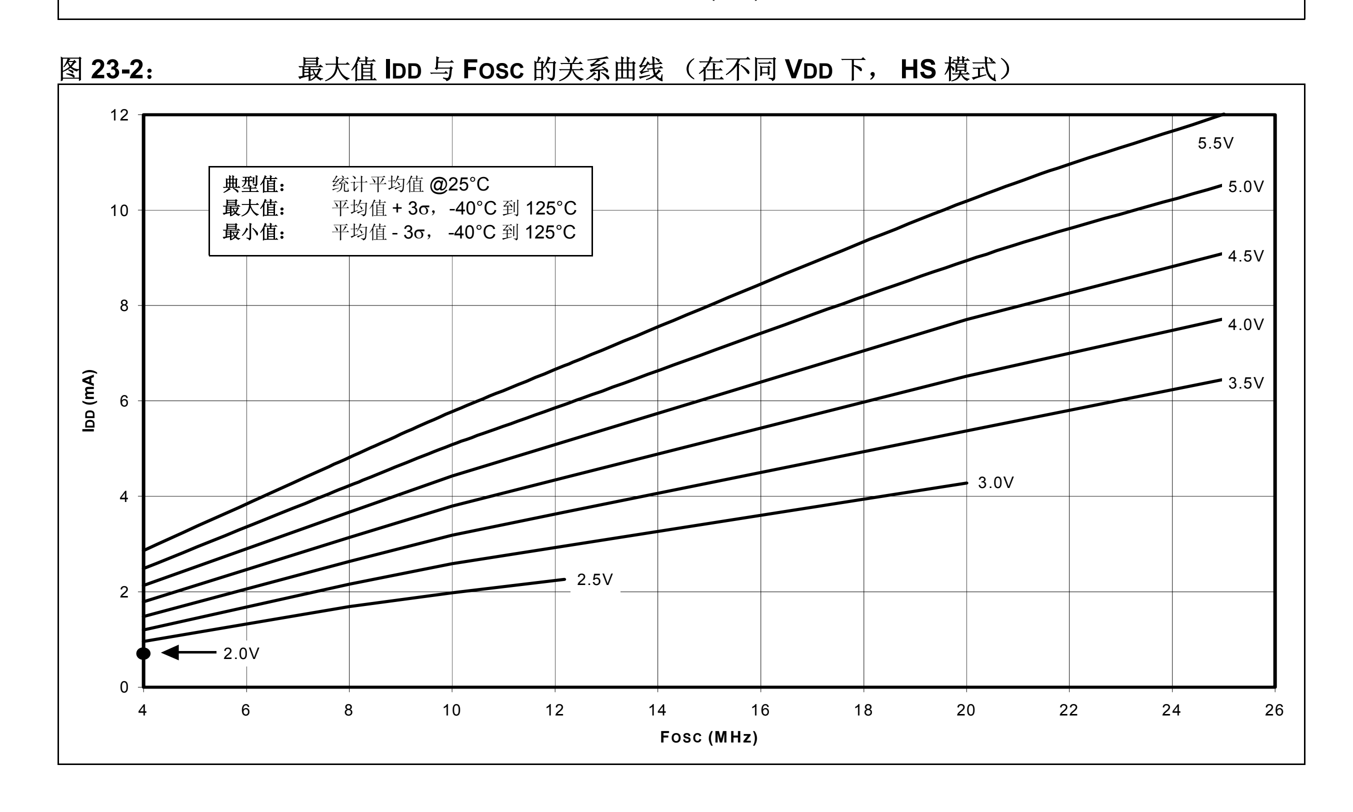


图 23-3： 典型 I_{DD} 与 F_{OSC} 的关系曲线（在不同 V_{DD} 下， HS/PLL 模式）

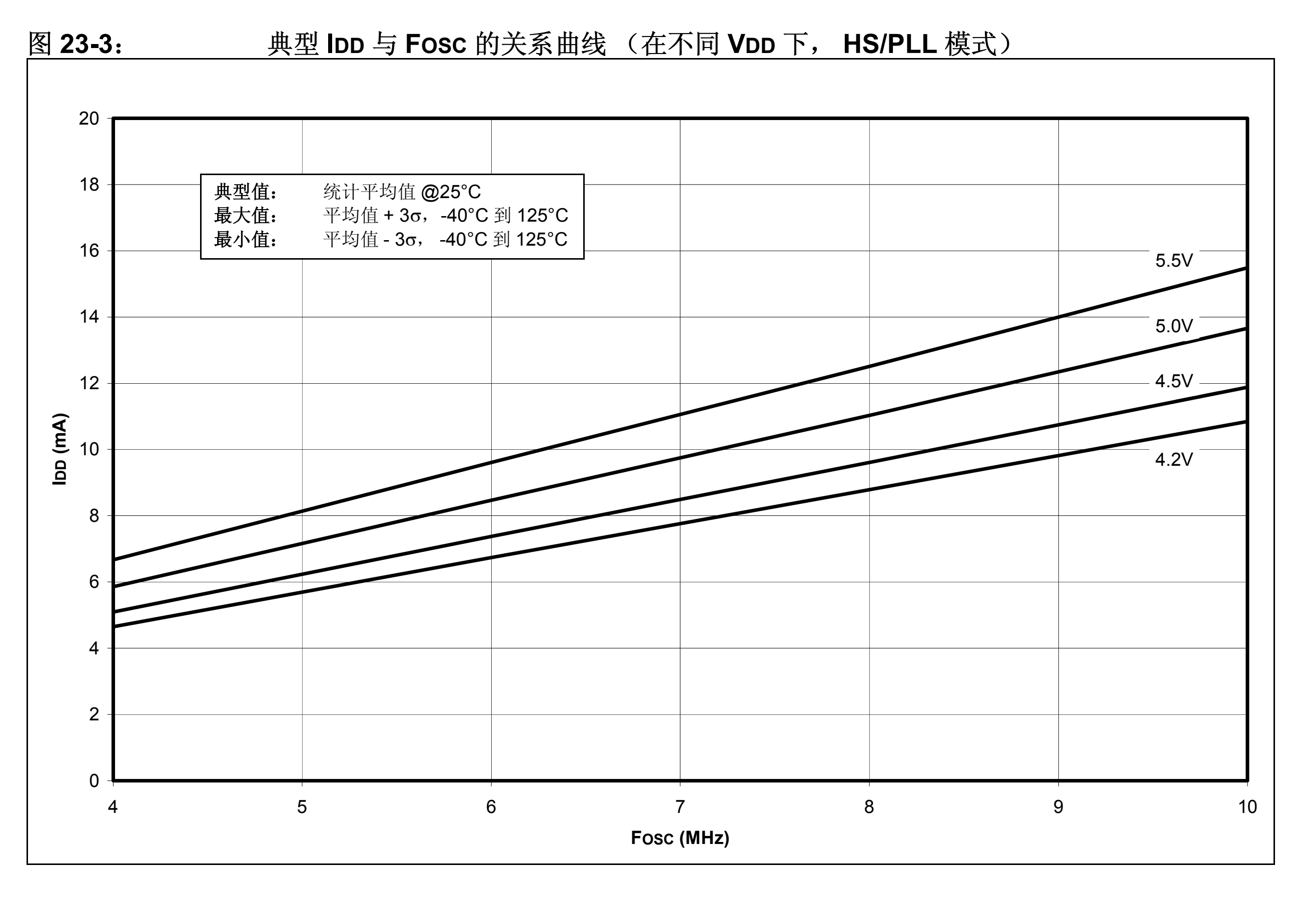


图 23-4： 最大 I_{DD} 与 F_{OSC} 的关系曲线（在不同 V_{DD} 下， HS/PLL 模式）

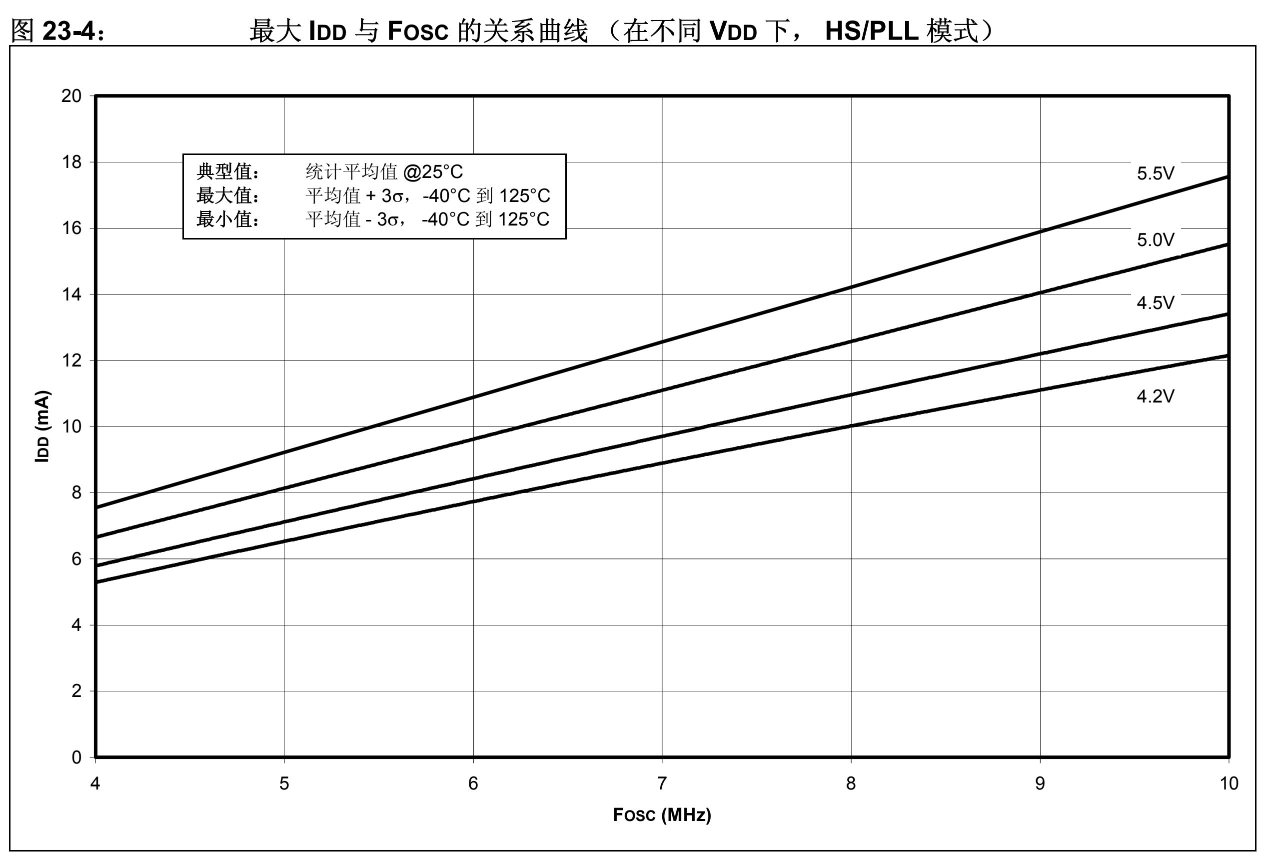


图 23-5：　　典型 IDD 与 FOSC 的关系曲线 （在不同 VDD 下， XT 模式）

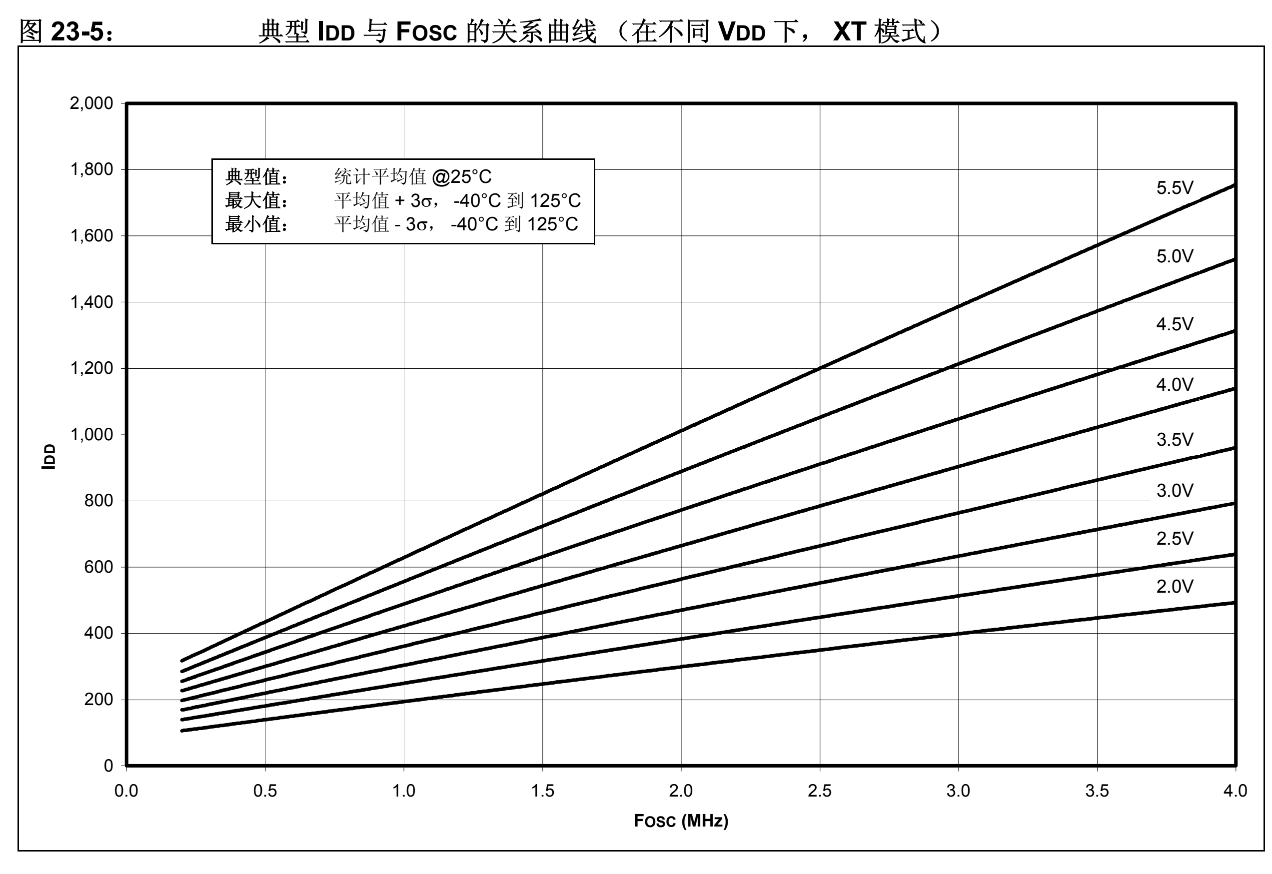


图 23-6：　　最大 IDD 与 FOSC 的关系曲线 （在不同 VDD 下， XT 模式）

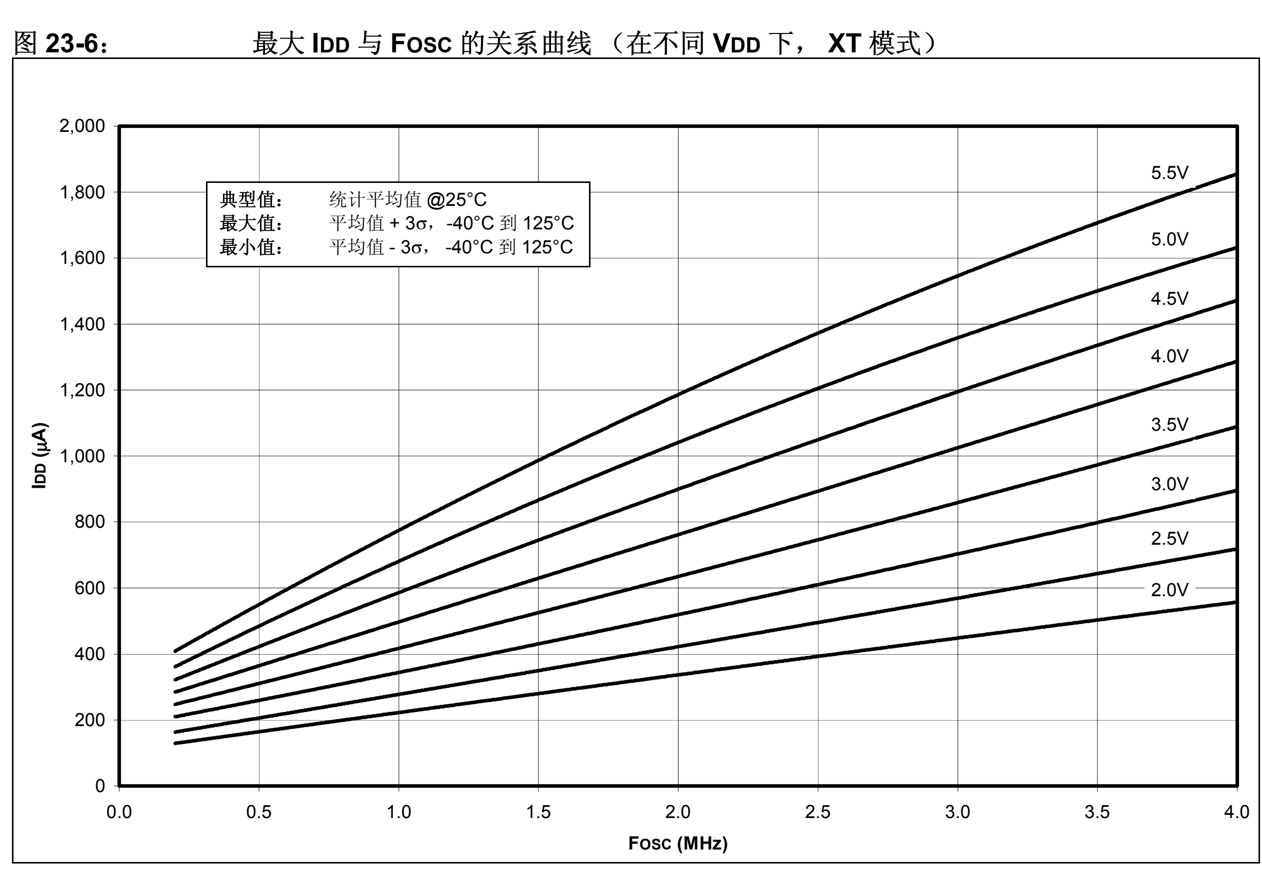


图 23-7：　　典型 IDD 与 FOSC 的关系曲线（在不同 VDD 下， LP 模式）

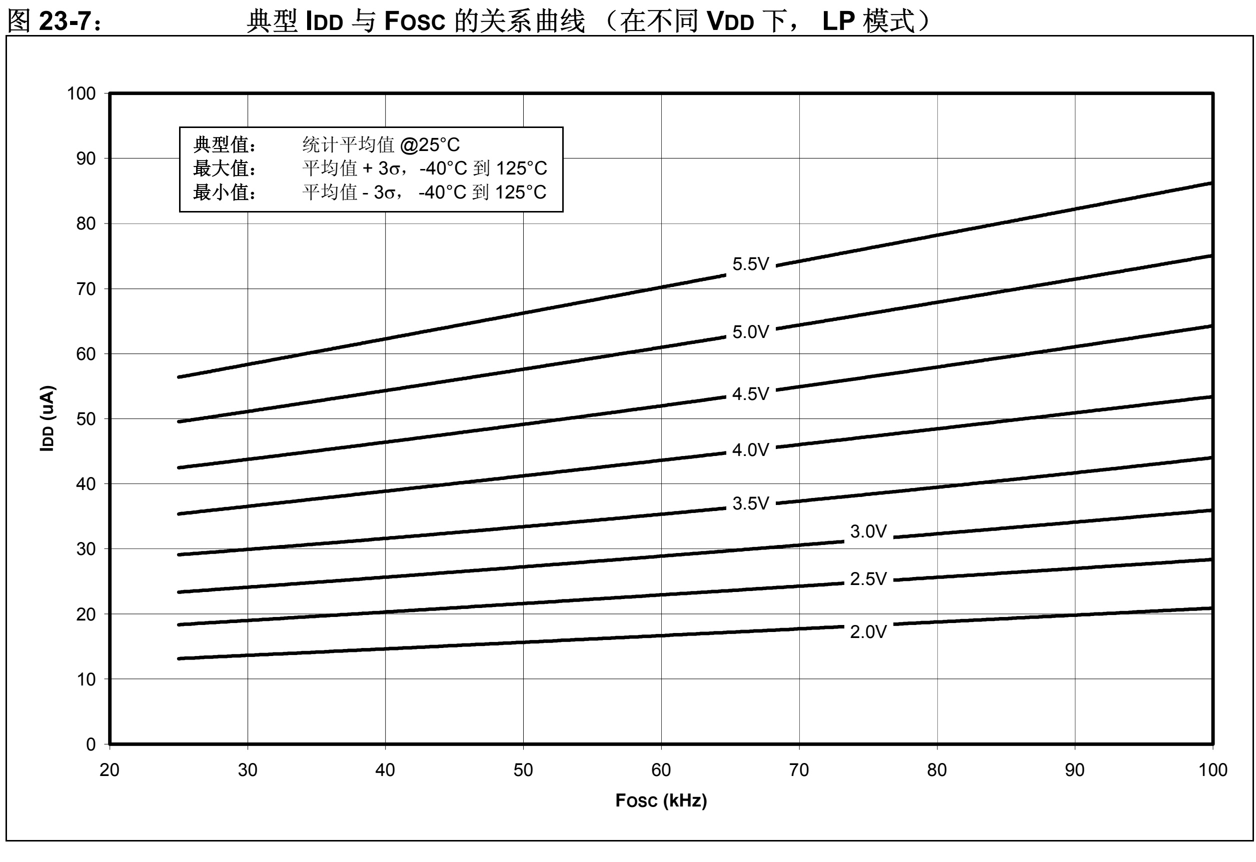


图 23-8：　　最大 IDD 与 FOSC 的关系曲线（在不同 VDD 下， LP 模式）

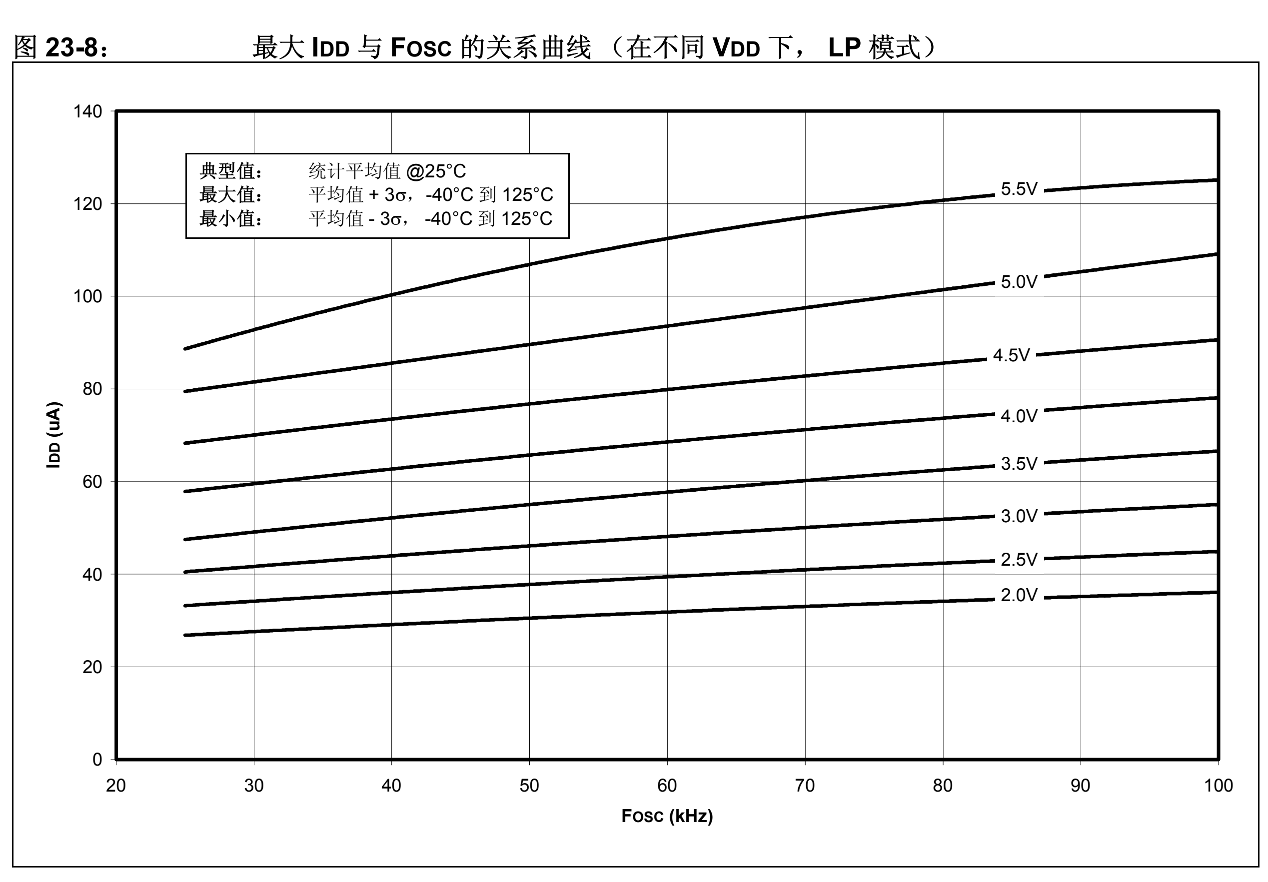


图 23-9：　　典型 IDD 与 FOSC 的关系曲线（在不同 VDD 下， EC 模式）

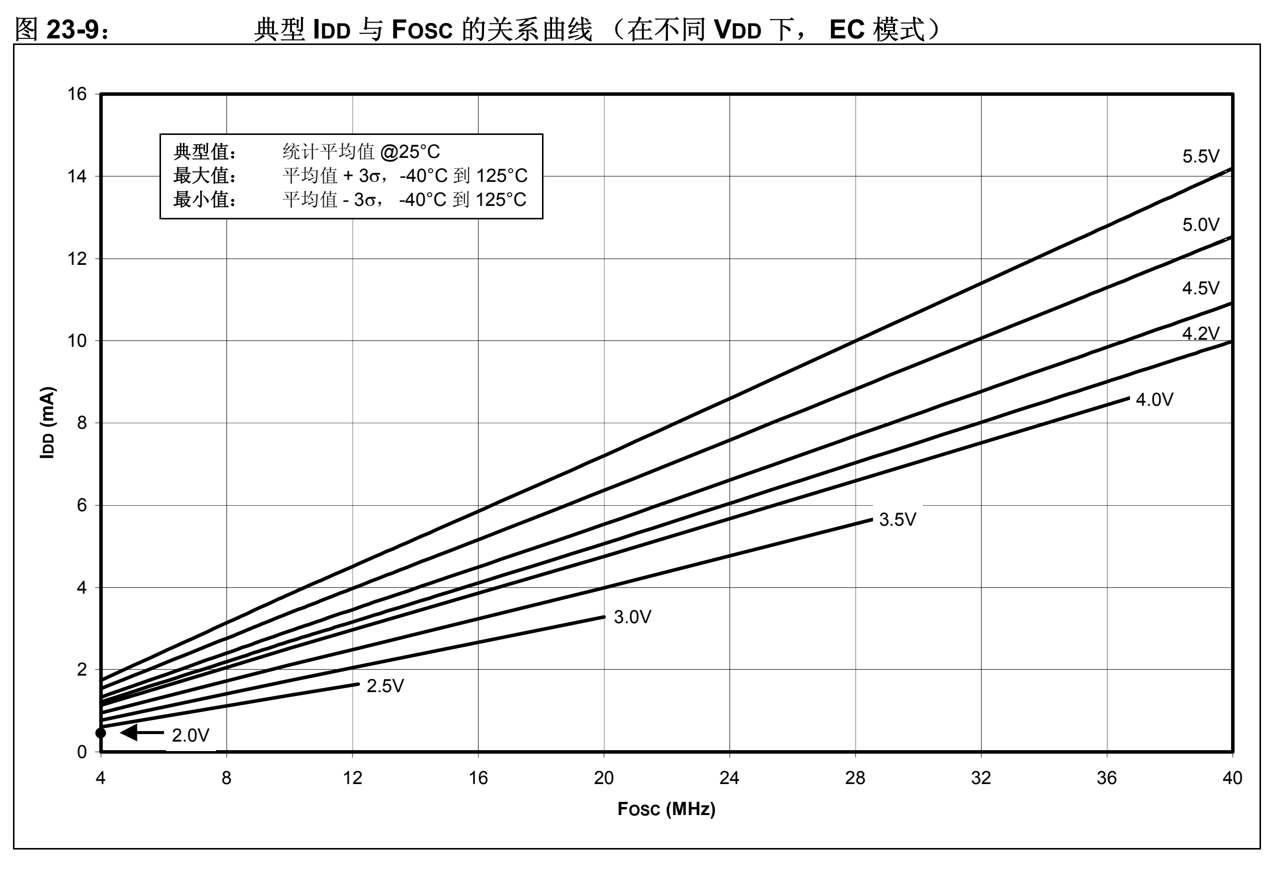


图 23-10：　　最大 IDD 与 FOSC 的关系曲线（在不同 VDD 下， EC 模式）

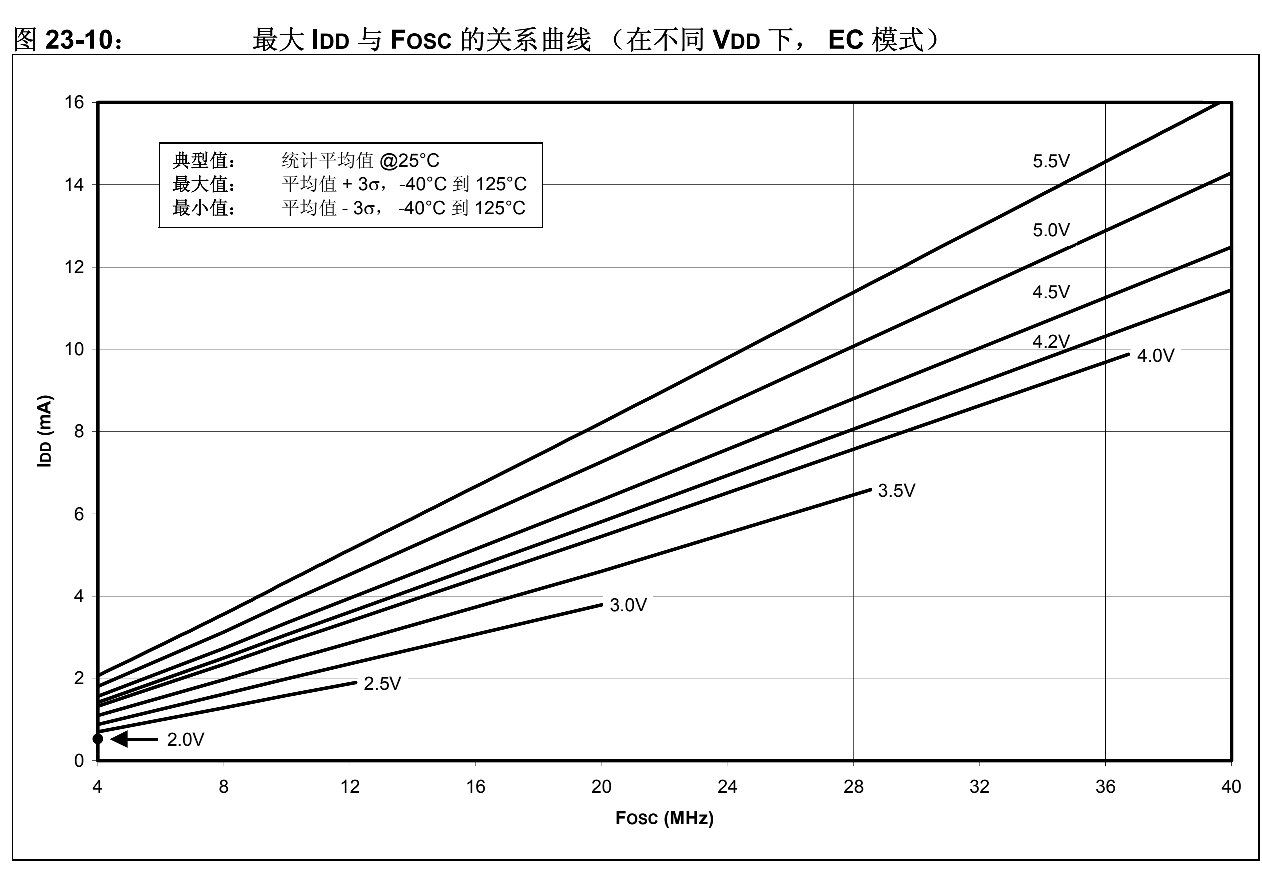


图 23-11： 典型和最大 IDD 与 VDD 的关系曲线
（TIMER1 作为主振荡器， 32.768 kHz， C1 和 C2=47 pF）

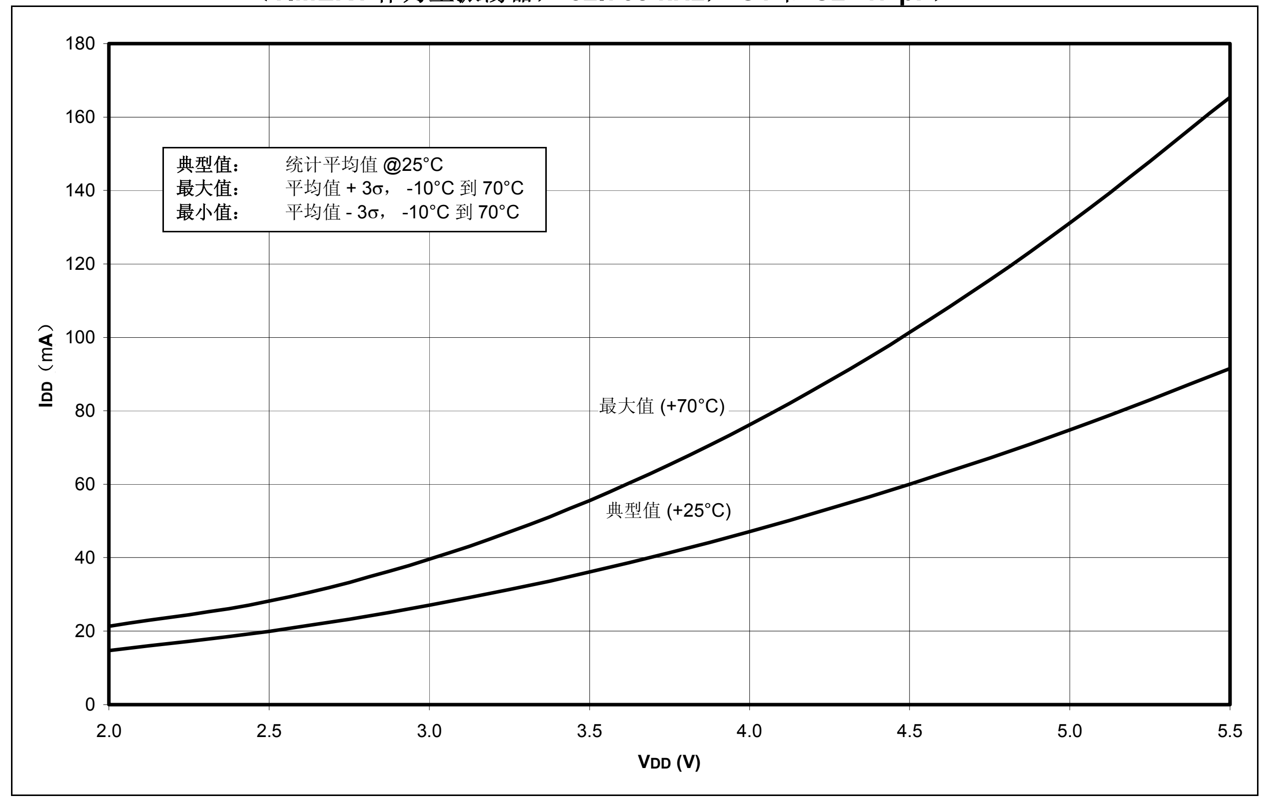


图 23-12： R 取不同值的情况下，平均 FOSC 与 VDD 的关系曲线
（RC 模式， C=20 pF， +25°C）

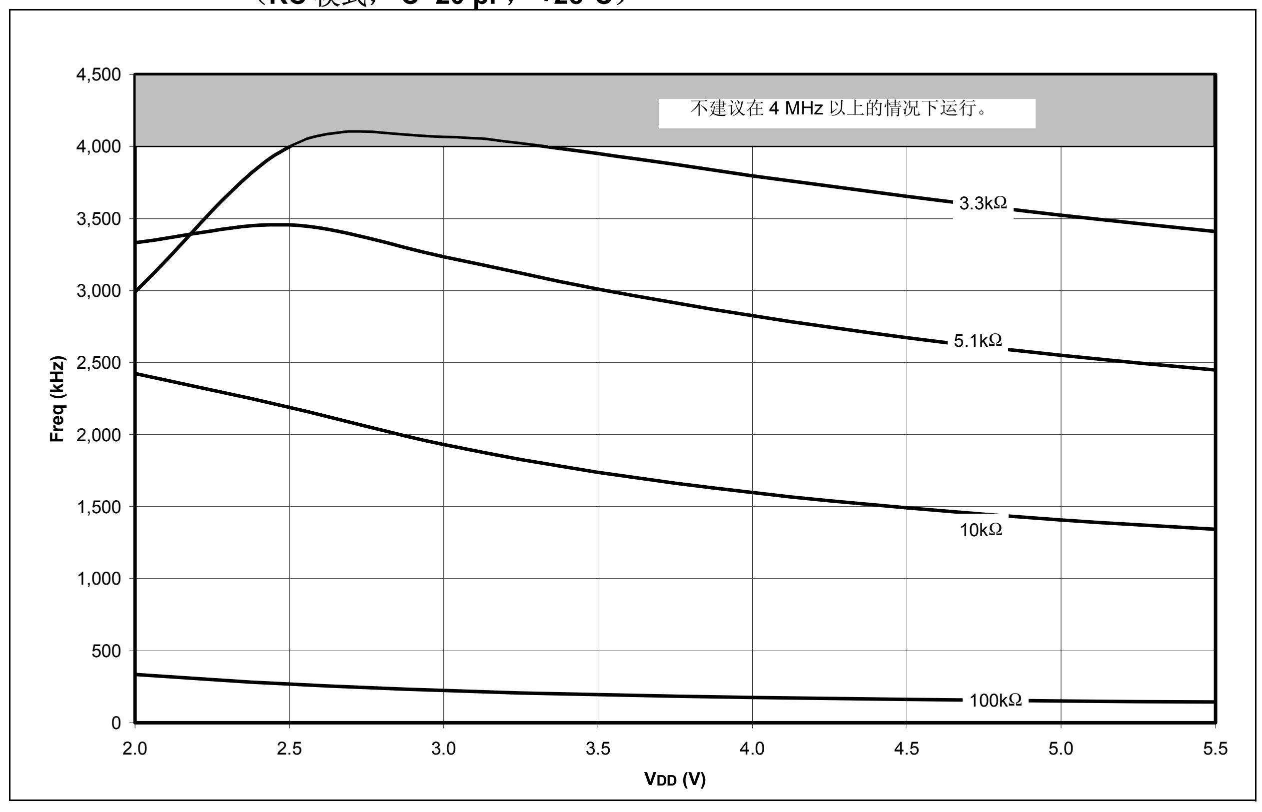


图 23-13： R 取不同值的情况下，平均 FOSC 与 VDD 的关系曲线（RC 模式， C=100 pF， +25°C）

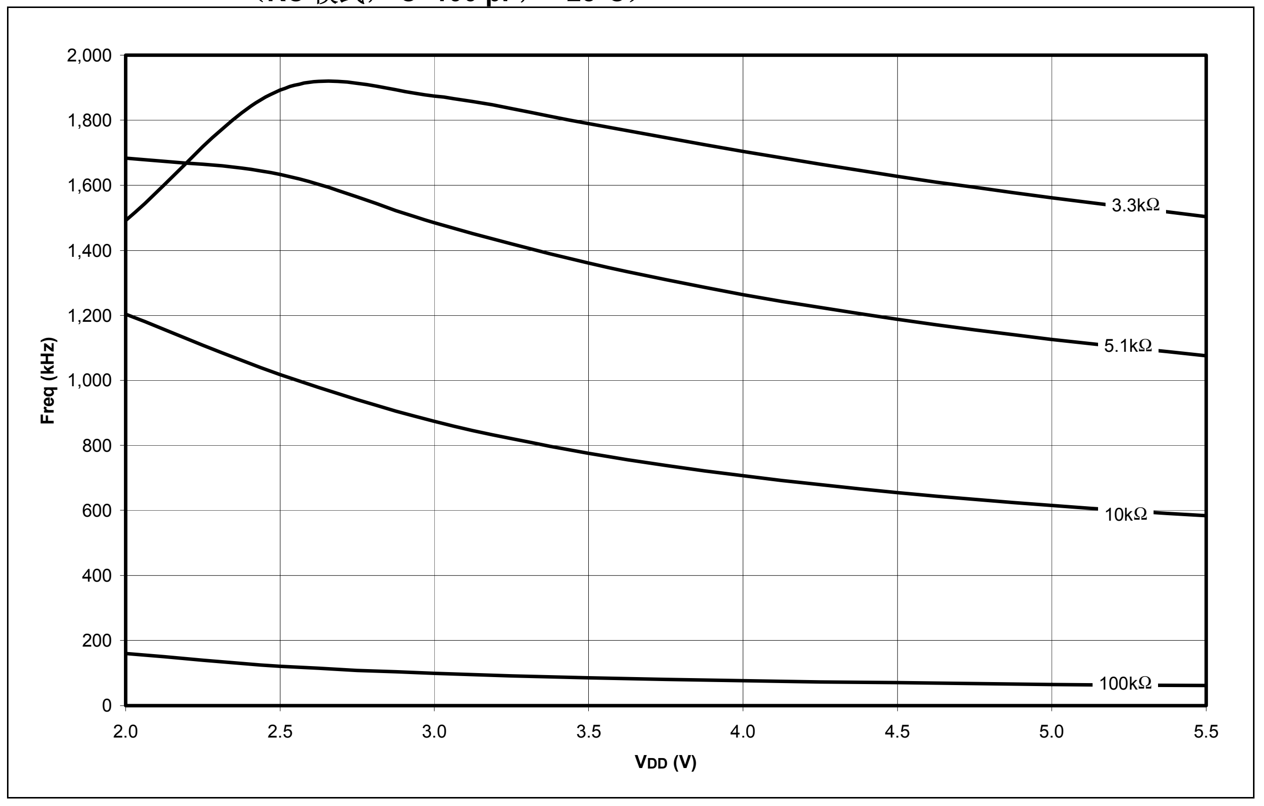


图 23-14： R 取不同值的情况下，平均 FOSC 与 VDD 的关系曲线（RC 模式， C=300 pF， +25°C）

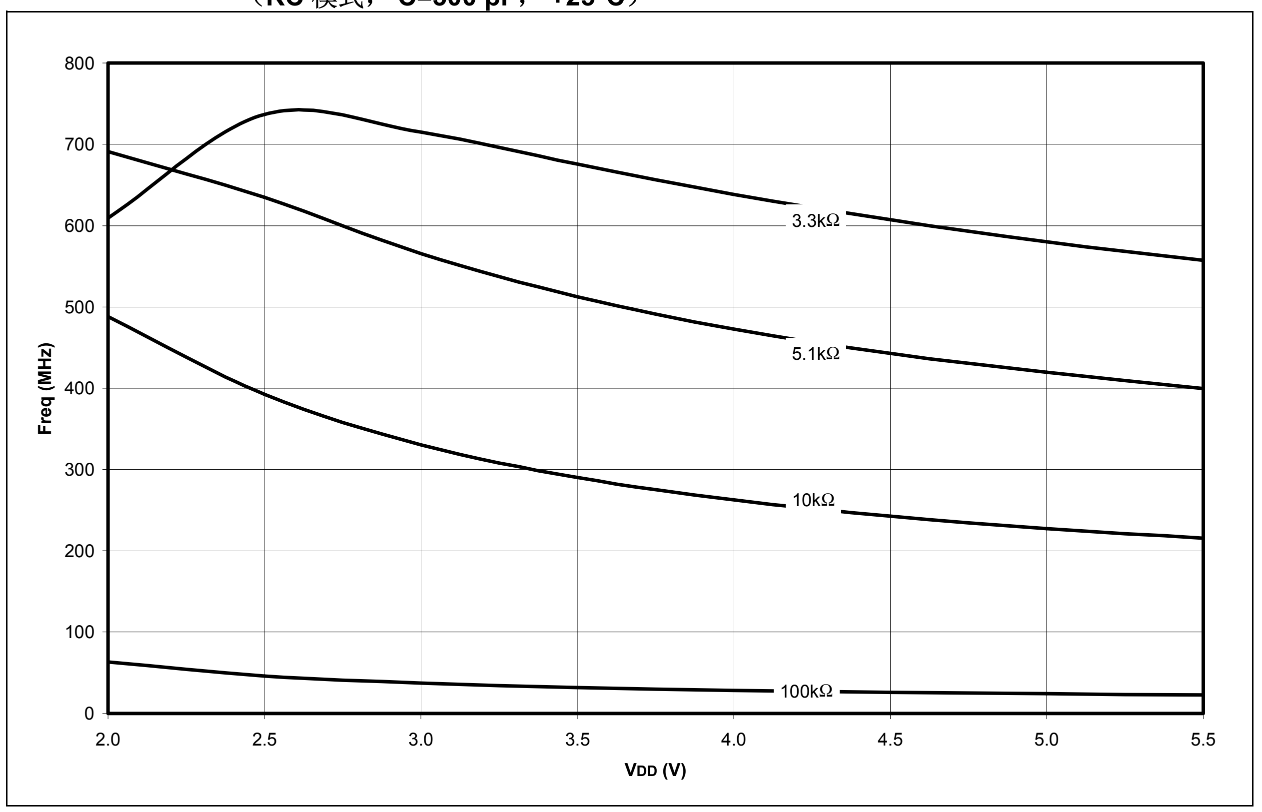


图 23-15： IPD 与 VDD 的关系曲线， -40°C 到 +125°C （休眠模式，禁止所有外设）

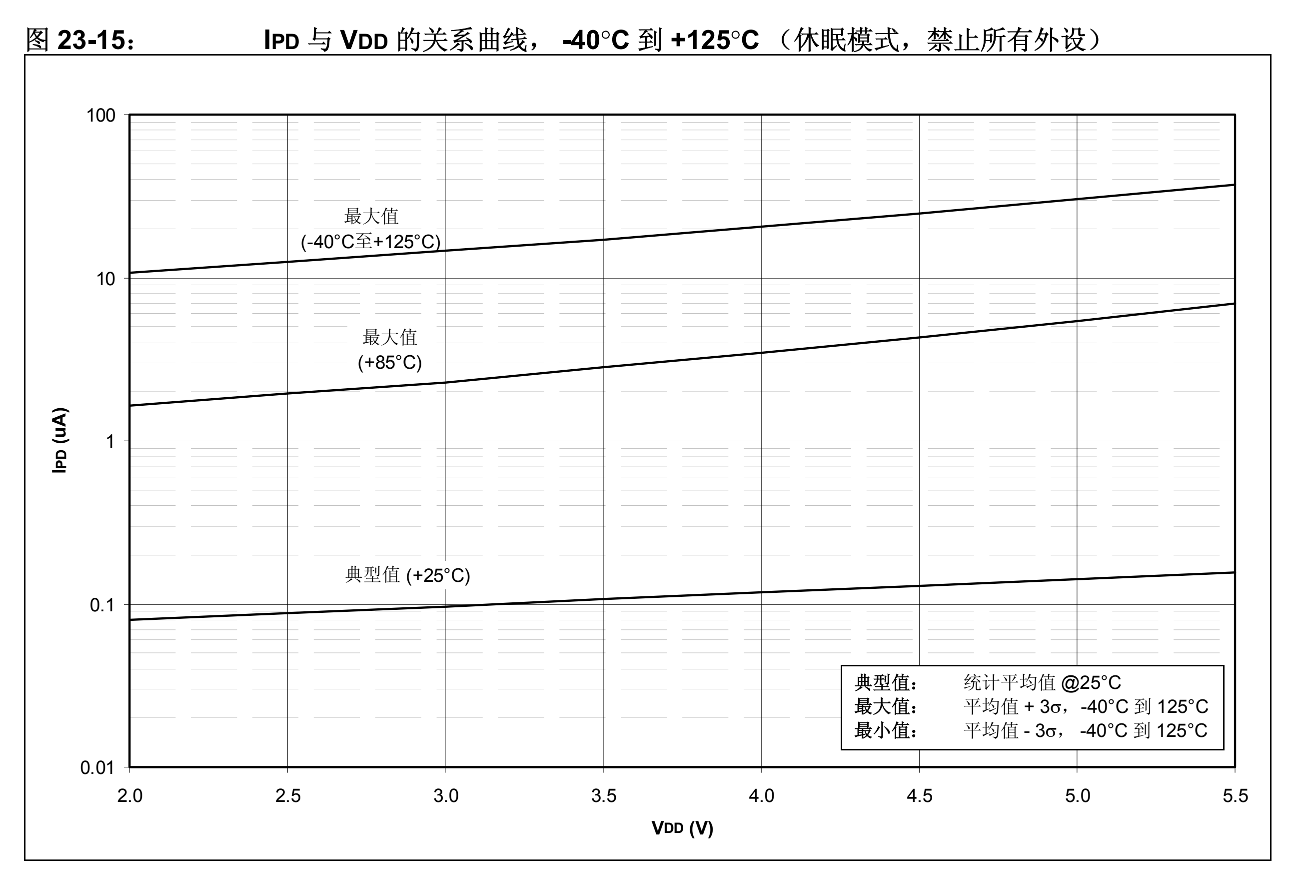


图 23-16： ΔIBOR 与 VDD 的关系曲线（在不同温度下， BOR 使能， VBOR= 2.00 - 2.16V）

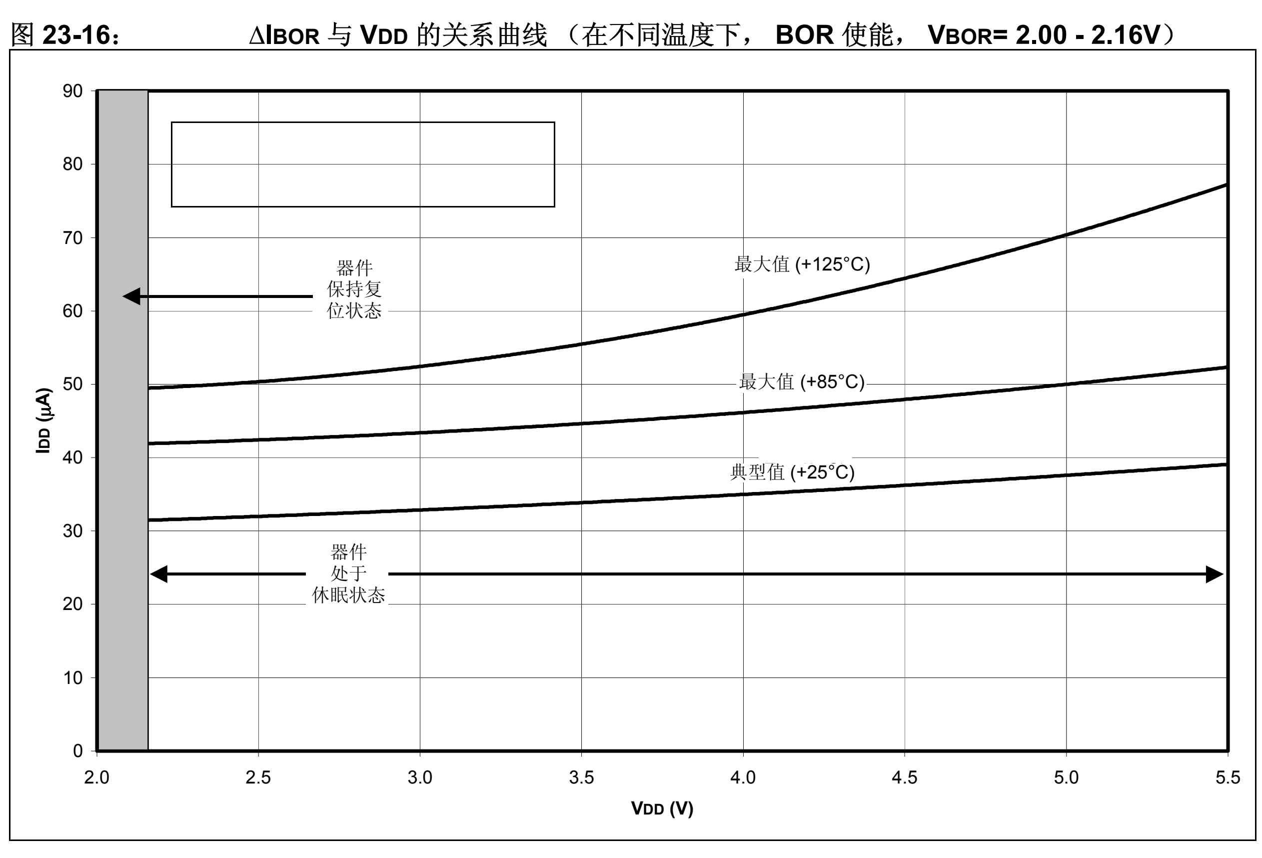


图 23-17： 典型和最大 ΔITMR1 与 VDD 的关系曲线 （在不同温度下， -10°C 到 +70°C，带振荡器的 TIMER1， XTAL=32 kHz， C1 和 C2=47 pF）

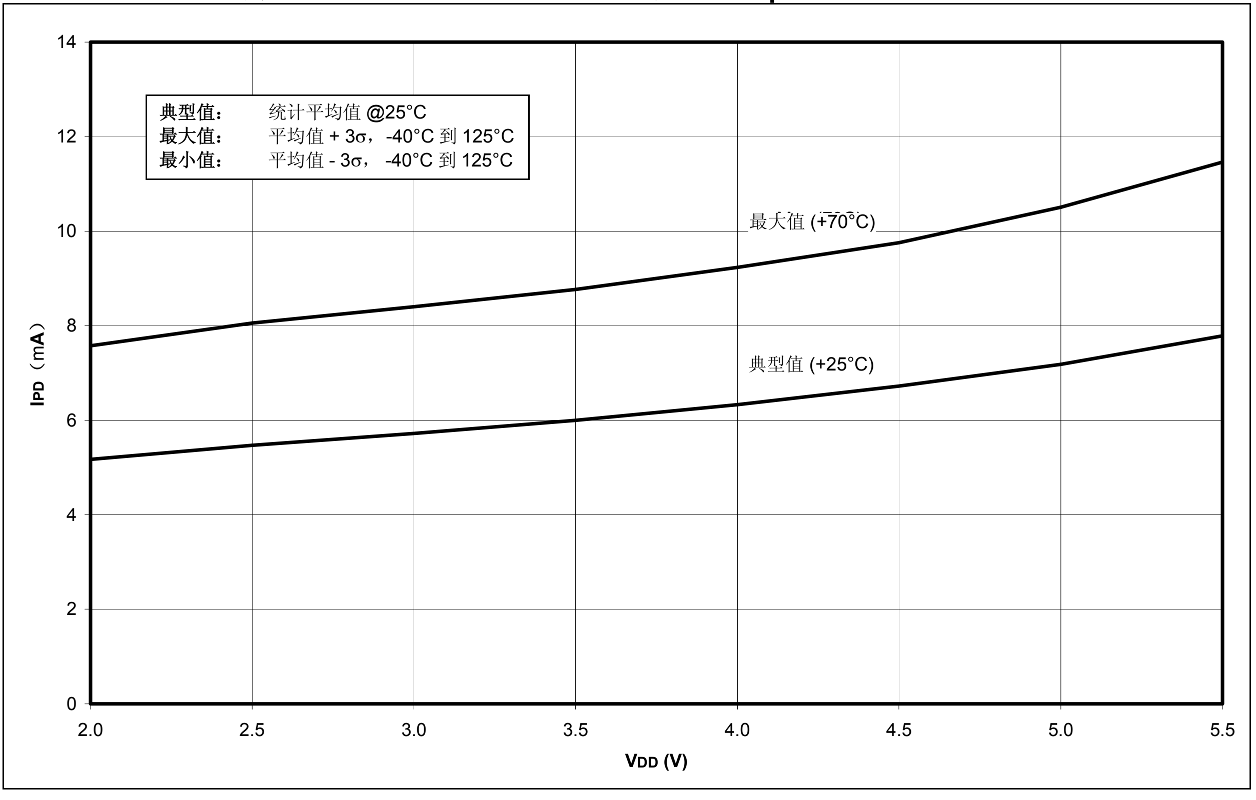


图 23-18： 典型和最大 ΔIWDT 与 VDD 的关系曲线 （在不同温度下， WDT 使能）

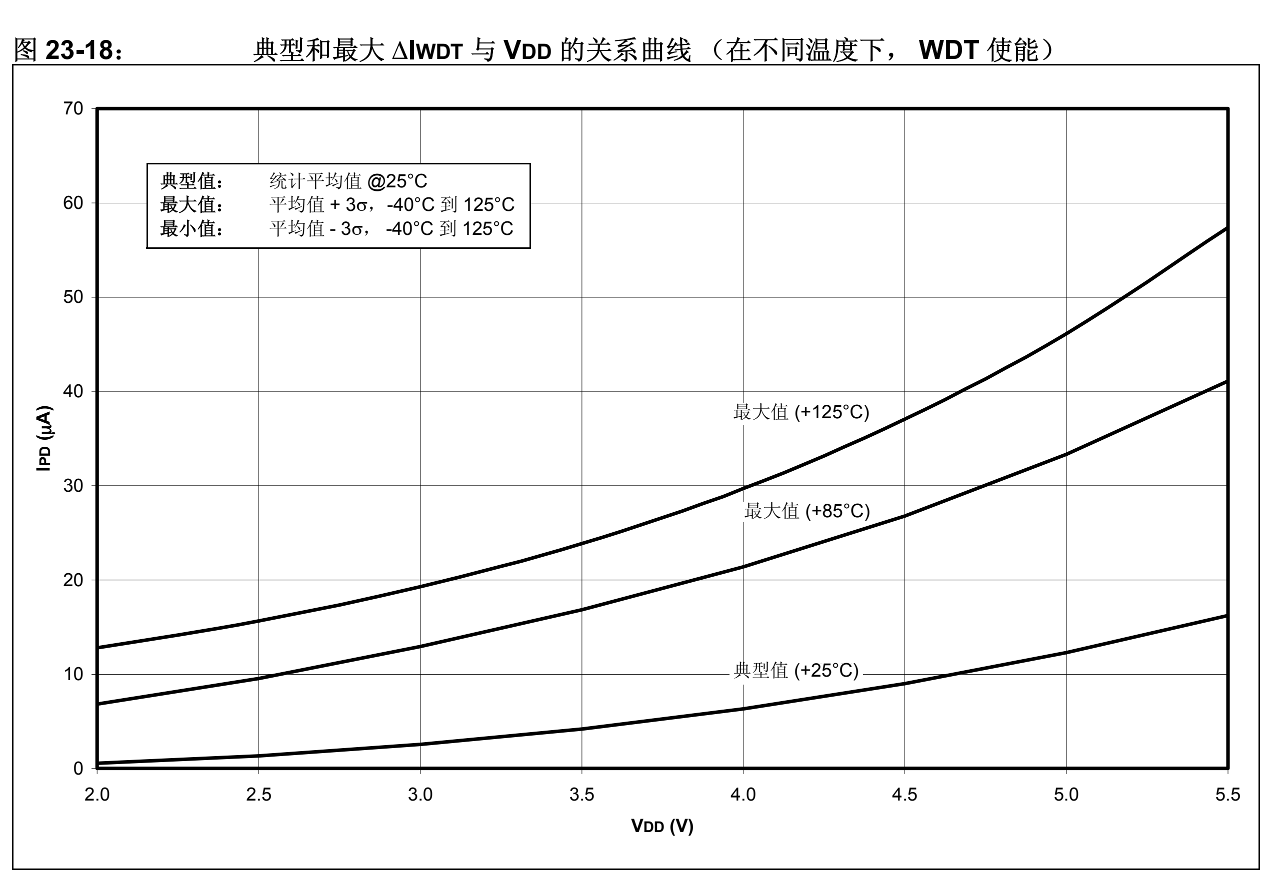


图 23-19： 典型、最小和最大 WDT 周期与 VDD 的关系曲线 （-40°C 到 +125°C）

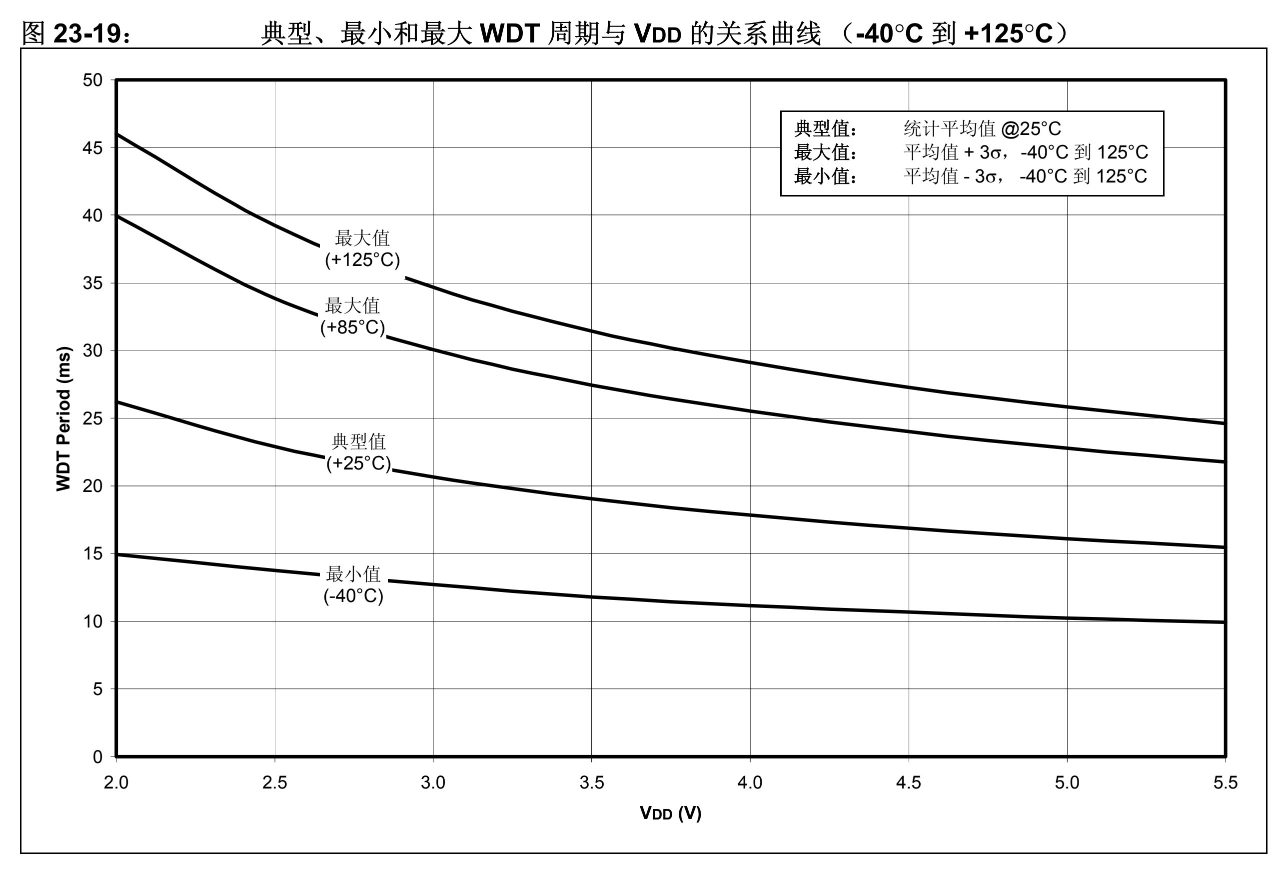


图 23-20： ΔILVD 与 VDD 的关系曲线 （在不同温度下， LVD 使能， VLVD = 4.5 - 4.78V）

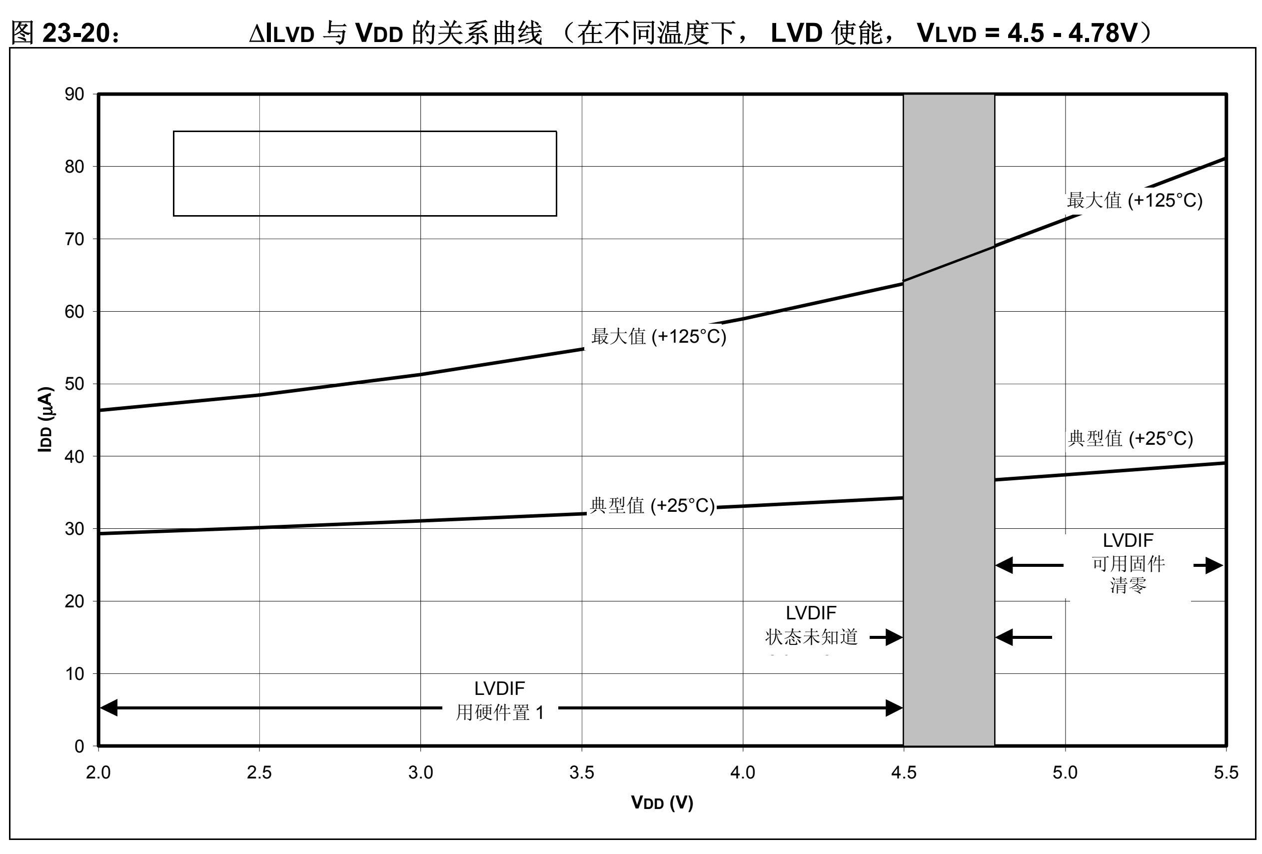


图 23-21： 典型、最小和最大 VOH 与 IOH 的关系曲线 （VDD = 5V， -40°C 到 +125°C）

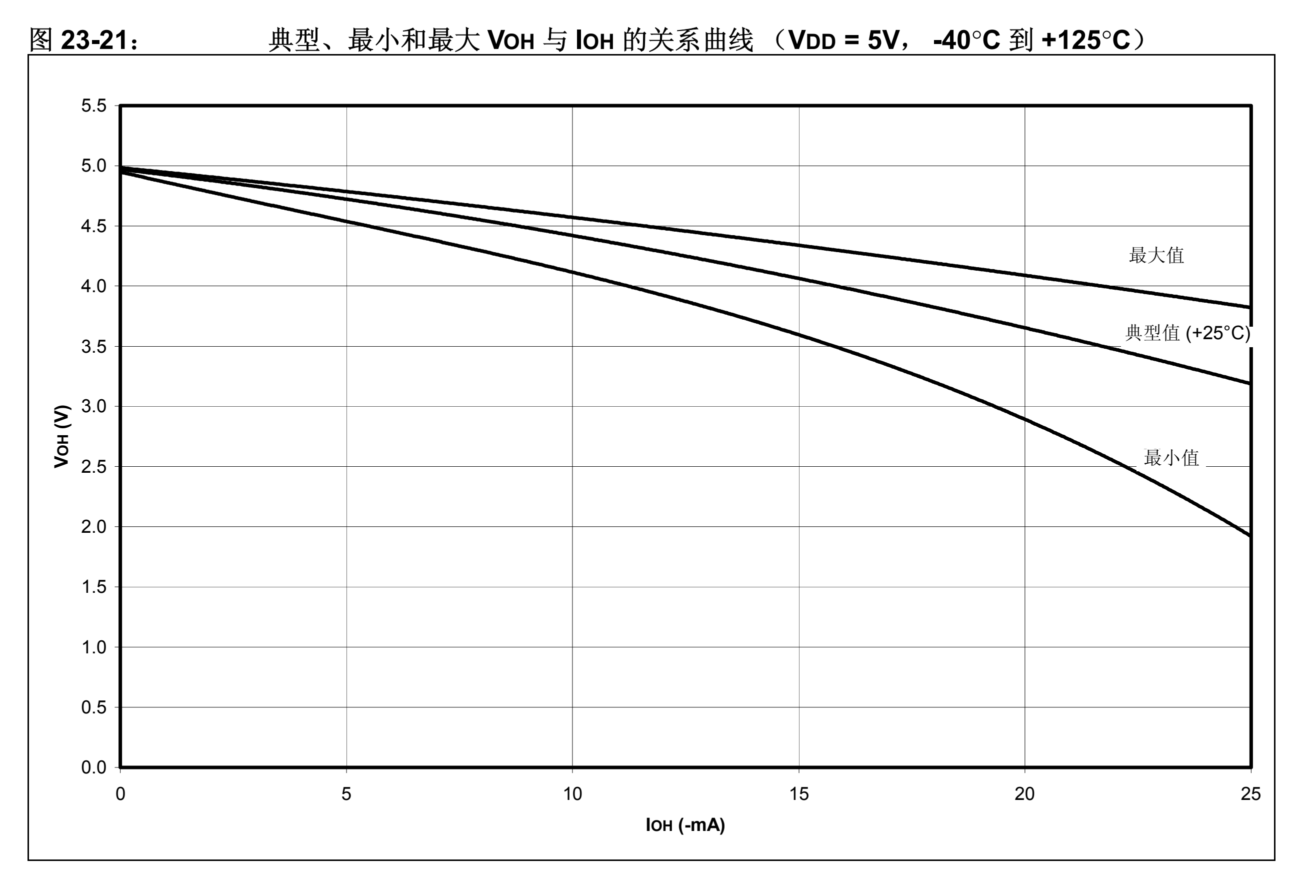


图 23-22： 典型、最小和最大 VOH 与 IOH 的关系曲线 （VDD = 3V， -40°C 到 +125°C）

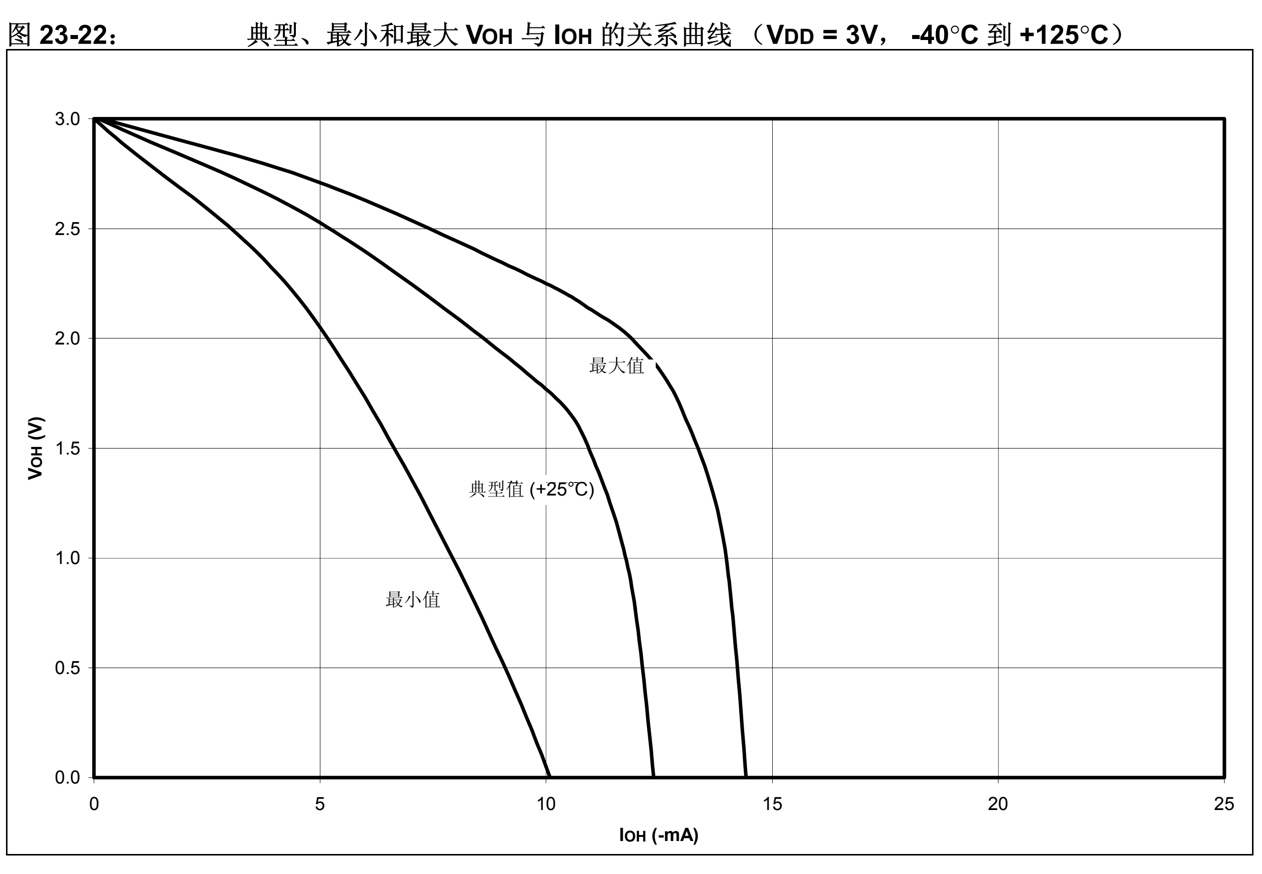


图 23-23： 典型和最大 VOL 与 IOL 的关系曲线 （VDD =5V， -40°C 到 +125°C）

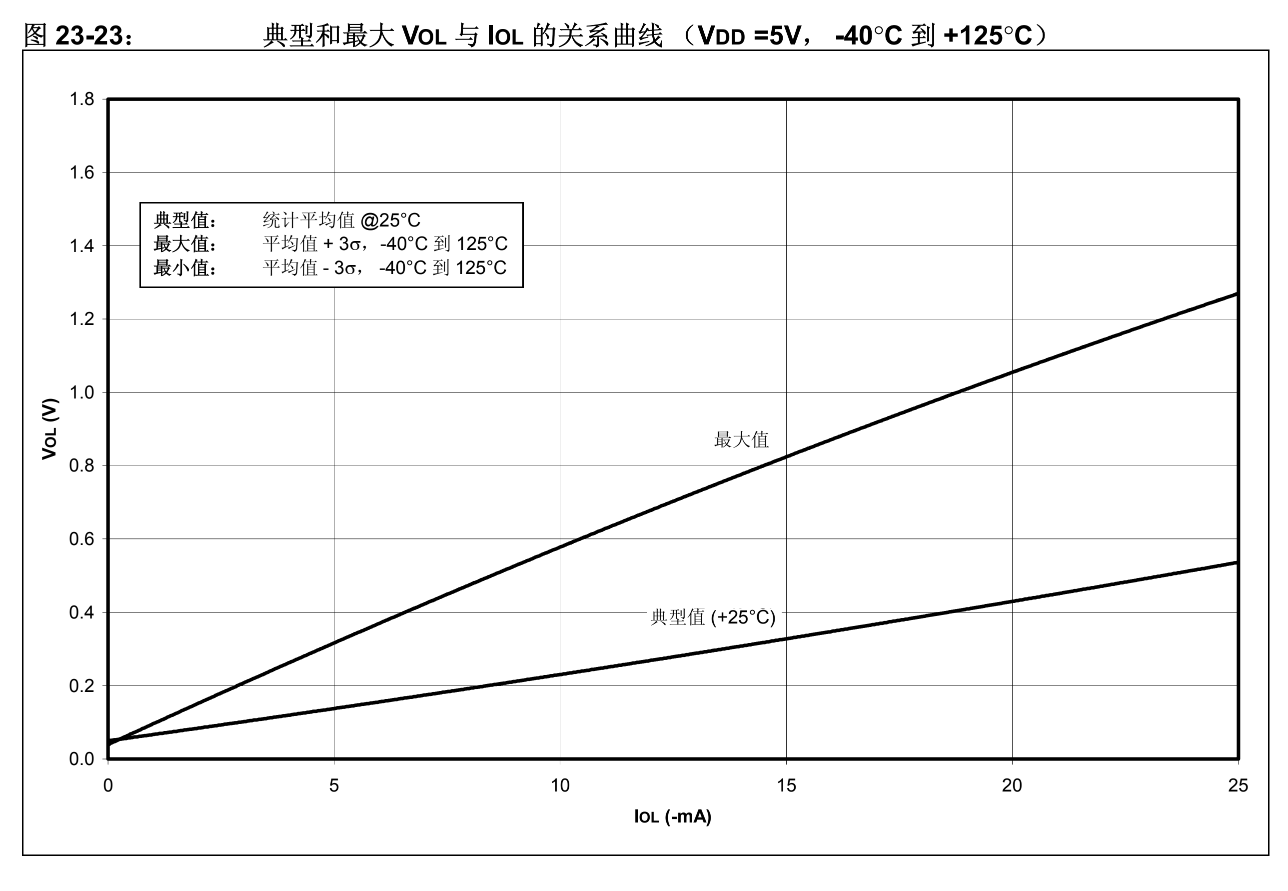


图 23-24： 典型和最大 VOL 与 IOL 的关系曲线 （VDD = 3V， -40°C 到 +125°C）

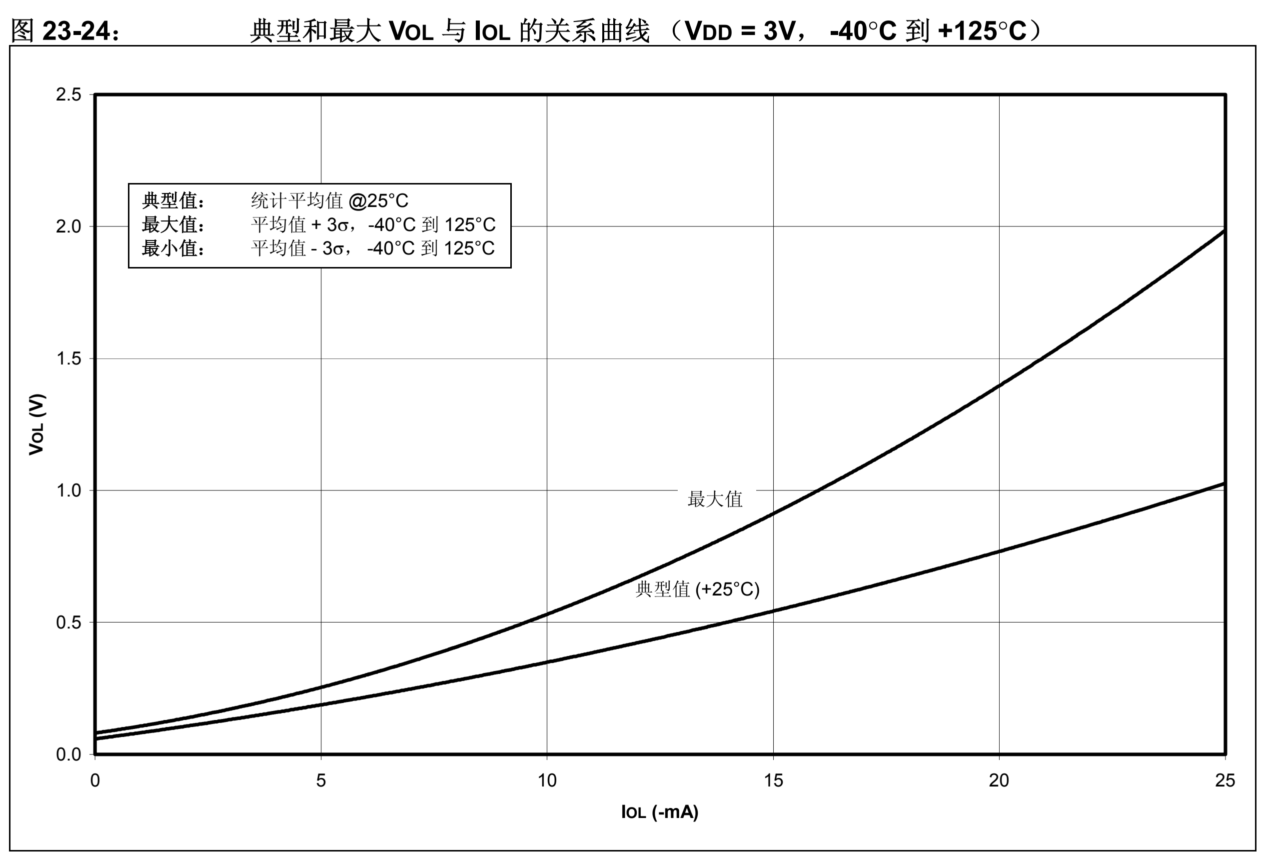


图 23-25： 最小和最大 VIN 与 VDD 的关系曲线 （ST 输入， -40°C 到 +125°C）

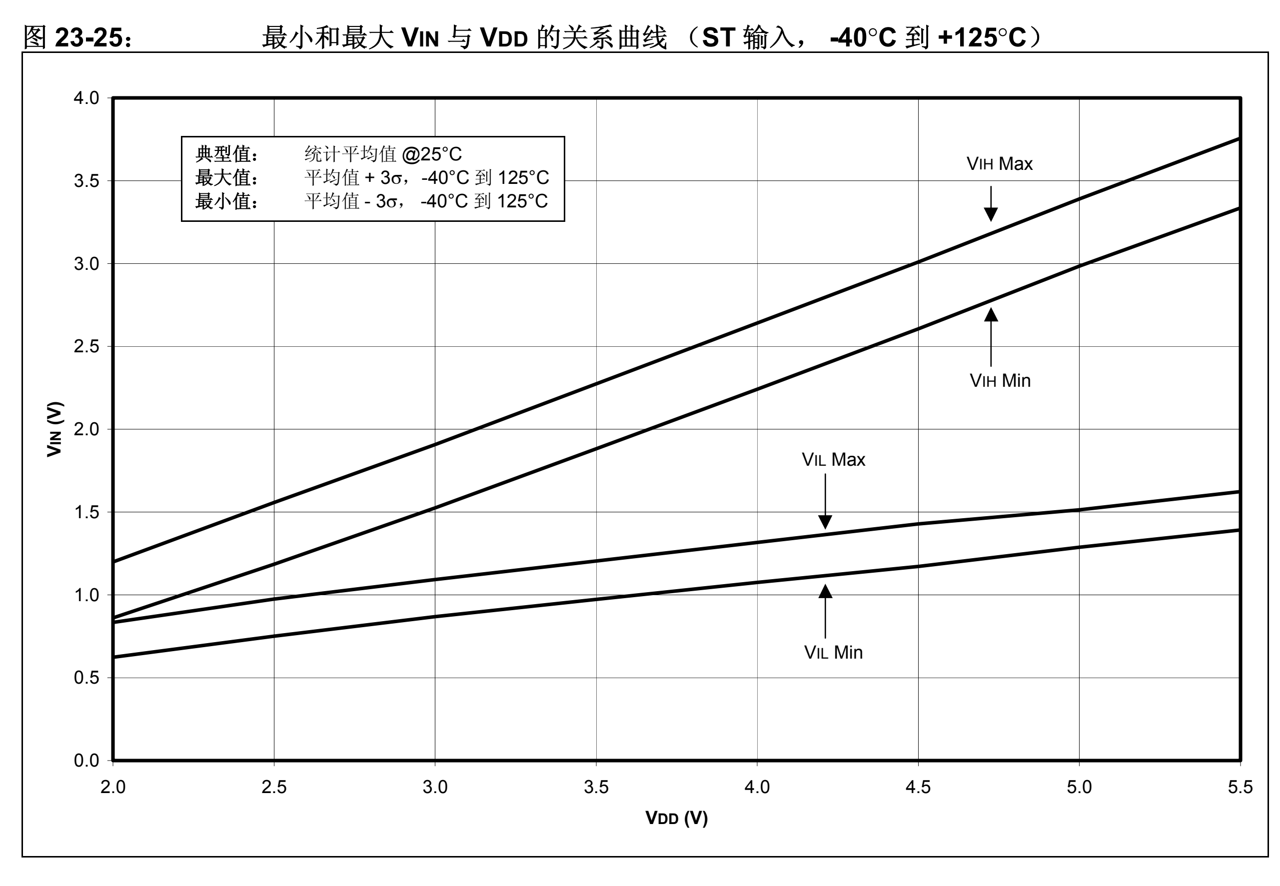


图 23-26： 最小和最大 VIN 与 VDD 的关系曲线 （TTL 输入， -40°C 到 +125°C）

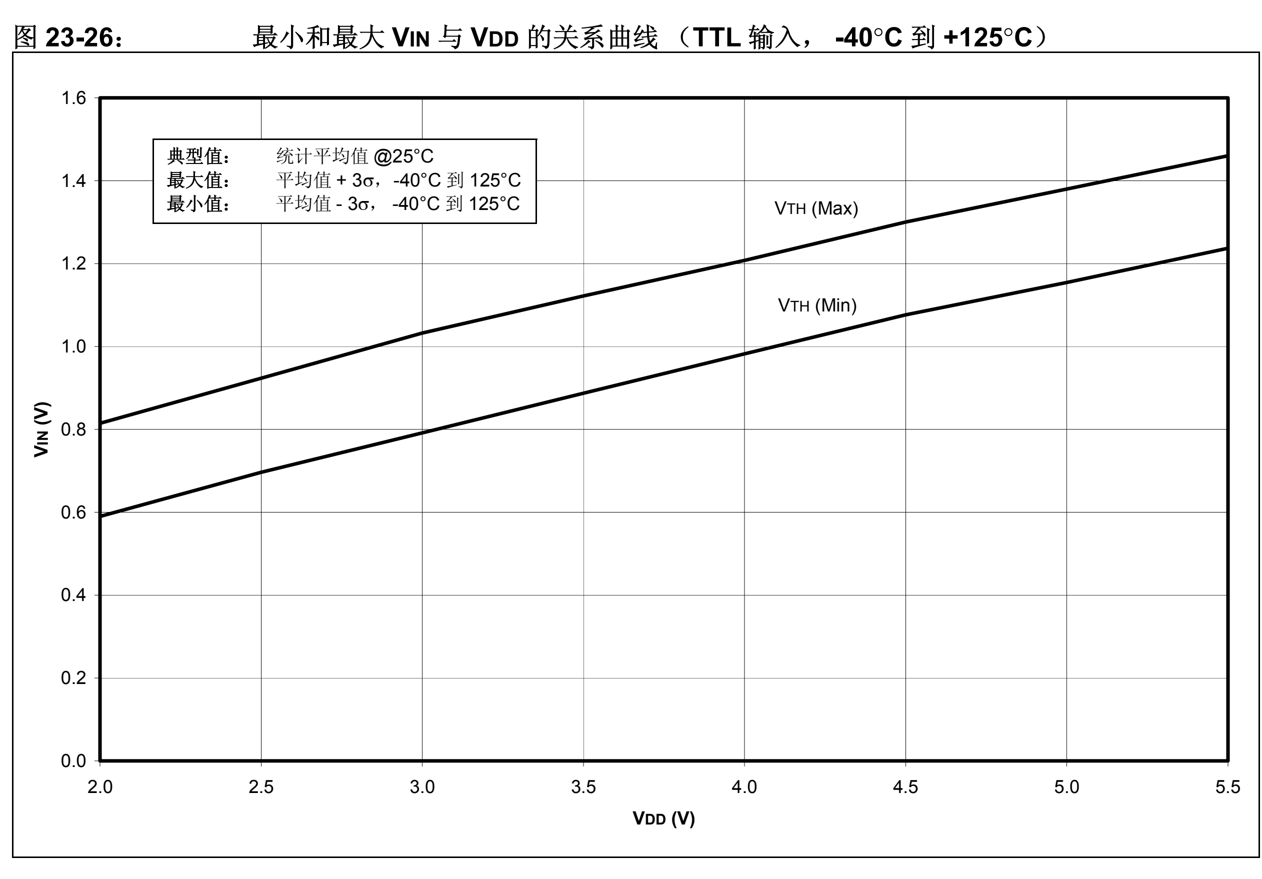


图 23-27： 最小和最大 VIN 与 VDD 的关系曲线（I^2C 输入， -40°C 到 +125°C）

典型值： 统计平均值 @25°C
最大值： 平均值 + 3σ， -40°C 到 125°C
最小值： 平均值 - 3σ， -40°C 到 125°C

VIH Max
VILMax
VIH Min
VIL Min
VIN (V)
VDD (V)

图 23-28： A/D 非线性与 VREFH 的关系曲线（VDD = VREFH， -40°C 到 +125°C）

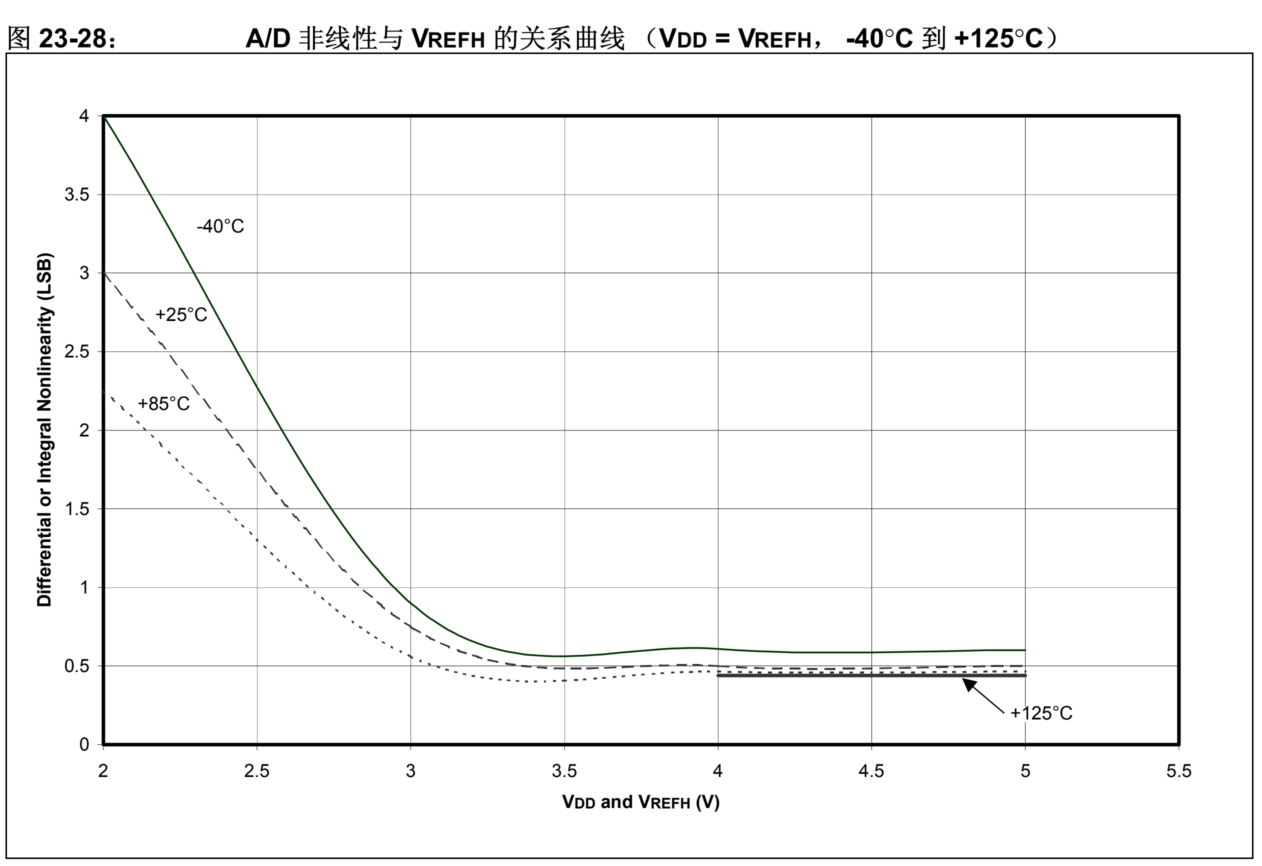
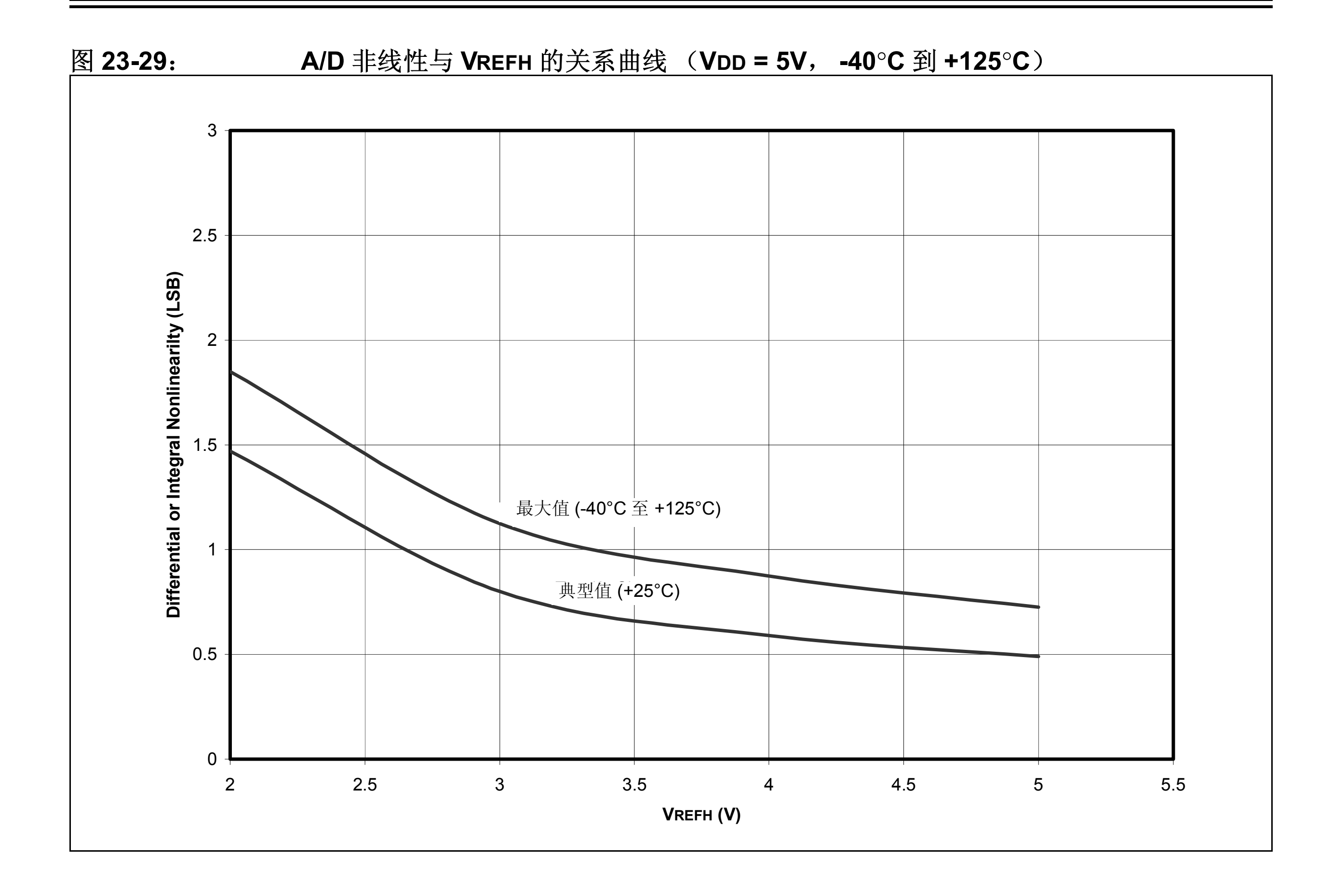


图 23-29： A/D 非线性与 VREFH 的关系曲线 （VDD = 5V， -40°C 到 +125°C）

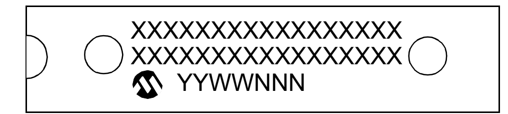


注：

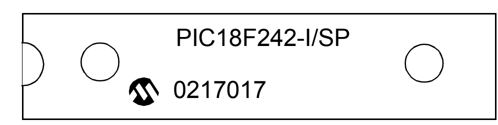
24.0 封装信息

24.1 封装标识信息

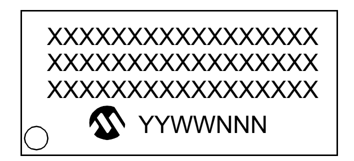
28 引脚 PDIP 封装（窄条 DIP）



示例



28 引脚 SOIC 封装



示例



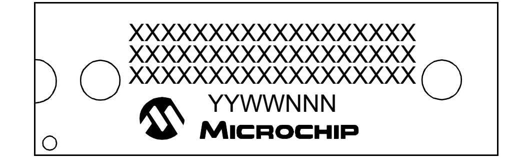
图注：		
	XX...X	客户信息 *
	Y	年份代码 （日历年的最后一位数字）
	YY	年份代码 （日历年的最后两位数字）
	WW	星期代码 （一月一日的星期代码为 “01”）
	NNN	以字母数字排序的追踪代码
	(e3)	雾锡 （Matte Tin， Sn）的 JEDEC 无铅标志 表示无铅封装。 JEDEC 无铅标志 （(e3)）标示于此种封装的外包装上。

注： 若 Microchip 器件编号未在一行中完全标出，它将换行继续标出。因此限制了客户特定信息的可用字符数。

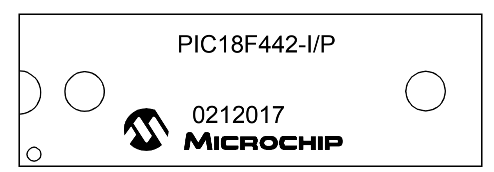
* 标准 PICmicro 器件标识包括 Microchip 器件编号、年份代码、星期代码和追踪代码。若 PICmicro 器件标识超出上述内容，需支付一定的附加费用。请向当地的 Microchip 销售办事处了解确认。对于 QTP 器件，任何特殊标记的费用都已包含在 QTP 价格中。

封装标识信息 （续）

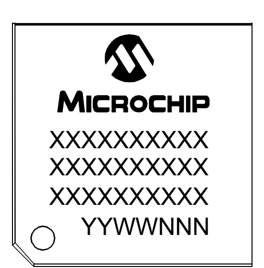
40 引脚 PDIP 封装



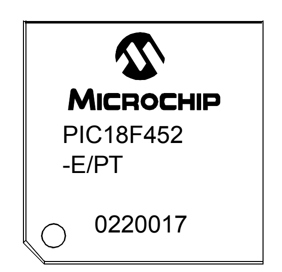
示例



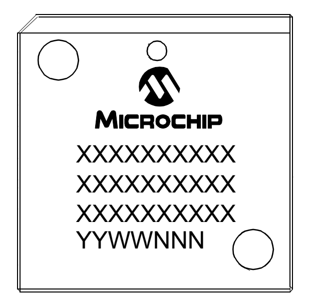
44 引脚 TQFP 封装



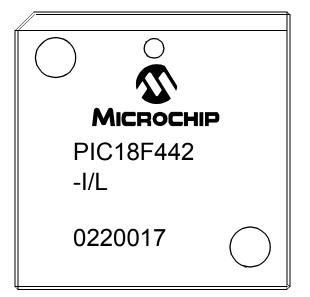
示例



44 引脚 PLCC 封装



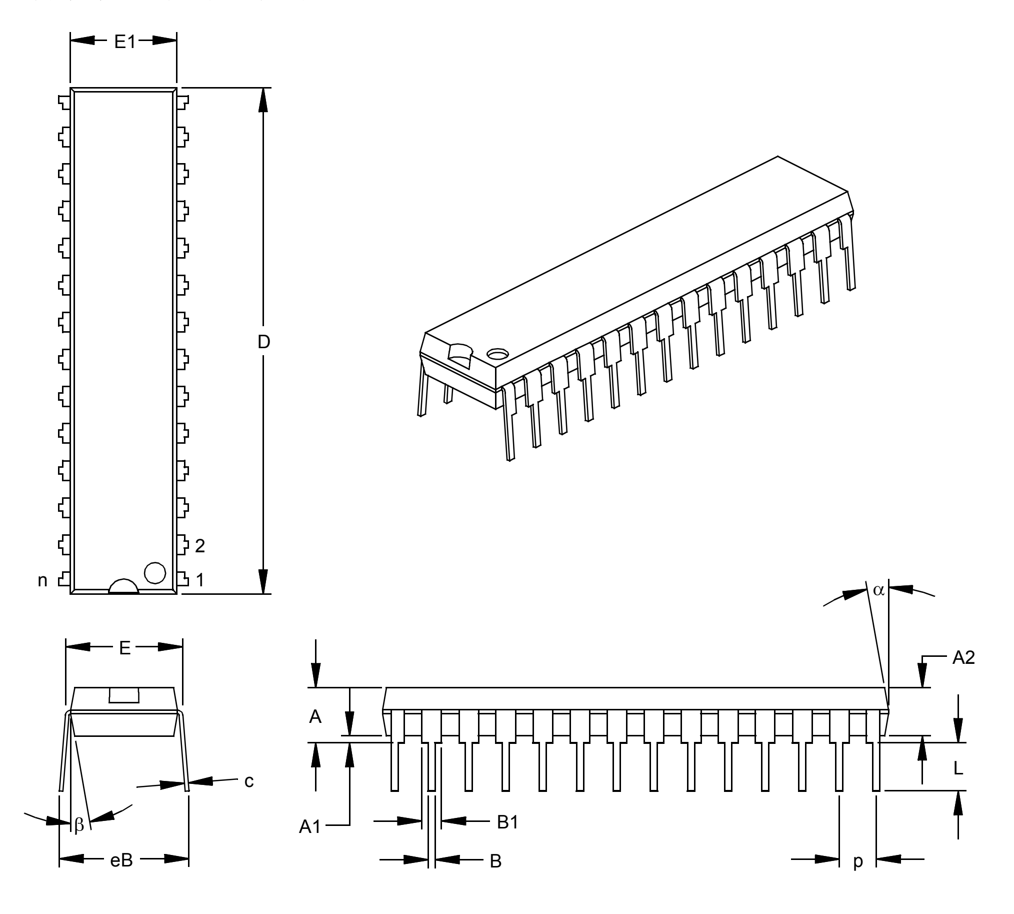
示例



24.2 封装详细信息

以下将介绍各种封装的技术细节。

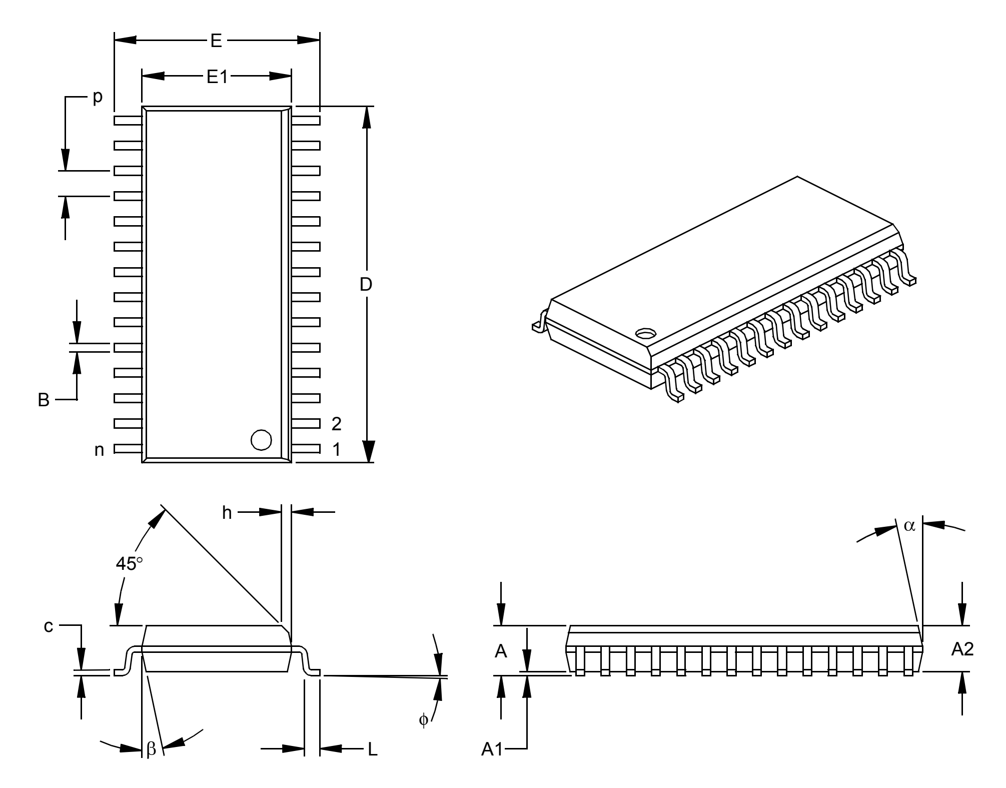
28 引脚塑料窄条双列直插式封装 （SP）— 300 mil （PDIP）



单位		英寸 *			毫米		
尺寸范围		最小值	标准值	最大值	最小值	标准值	最大值
引脚数	n		28			28	
引脚间距	p		.100			2.54	
顶端到固定面高度	A	.140	.150	.160	3.56	3.81	4.06
模制封装厚度	A2	.125	.130	.135	3.18	3.30	3.43
塑模底端到固定面高度	A1	.015			0.38		
肩到肩宽度	E	.300	.310	.325	7.62	7.87	8.26
塑模封装宽度	E1	.275	.285	.295	6.99	7.24	7.49
总长度	D	1.345	1.365	1.385	34.16	34.67	35.18
引脚尖到固定面高度	L	.125	.130	.135	3.18	3.30	3.43
引脚厚度	c	.008	.012	.015	0.20	0.29	0.38
引脚上部宽度	B1	.040	.053	.065	1.02	1.33	1.65
引脚下部宽度	B	.016	.019	.022	0.41	0.48	0.56
总排列间距 §	eB	.320	.350	.430	8.13	8.89	10.92
塑模顶部锥度	α	5	10	15	5	10	15
塑模底部锥度	β	5	10	15	5	10	15

* 控制参数
§ 重要特性参数
注:
尺寸 D 和 E1 不包括塑模毛边或突起。每侧的塑模毛边或突起不得超过 0.010 英寸（0.254 毫米）。
等同 JEDEC 规范: MO-095
图号 C04-070

28 引脚塑封小型封装 （SO）—宽条 300 mil （SOIC）

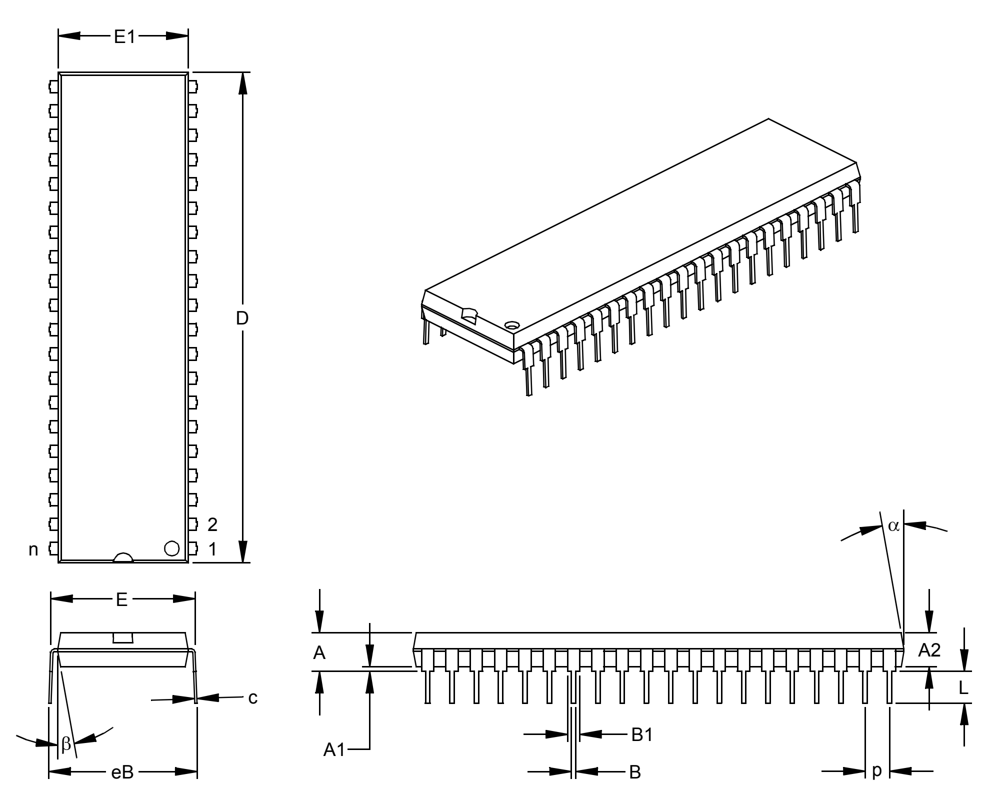


单位		英寸 *			毫米		
尺寸范围		最小值	标准值	最大值	最小值	标准值	最大值
引脚数	n		28			28	
引脚间距	p		.050			1.27	
顶端到固定面高度	A	.093	.099	.104	2.36	2.50	2.64
塑模封装厚度	A2	.088	.091	.094	2.24	2.31	2.39
悬空间隙 §	A1	.004	.008	.012	0.10	0.20	0.30
总宽度	E	.394	.407	.420	10.01	10.34	10.67
塑模封装宽度	E1	.288	.295	.299	7.32	7.49	7.59
总长度	D	.695	.704	.712	17.65	17.87	18.08
斜面投影距离	h	.010	.020	.029	0.25	0.50	0.74
底脚长度	L	.016	.033	.050	0.41	0.84	1.27
底脚倾斜角	φ	0	4	8	0	4	8
引脚厚度	c	.009	.011	.013	0.23	0.28	0.33
引脚宽度	B	.014	.017	.020	0.36	0.42	0.51
塑模顶部锥度	α	0	12	15	0	12	15
塑模底部锥度	β	0	12	15	0	12	15

* 控制参数
§ 重要特性参数

注：
尺寸 D 和 E1 不包括塑模毛边或突起。每侧的塑模毛边或突起不得超过 0.010 英寸（0.254 毫米）。
等同 JEDEC 规范：MS-013
图号 C04-052

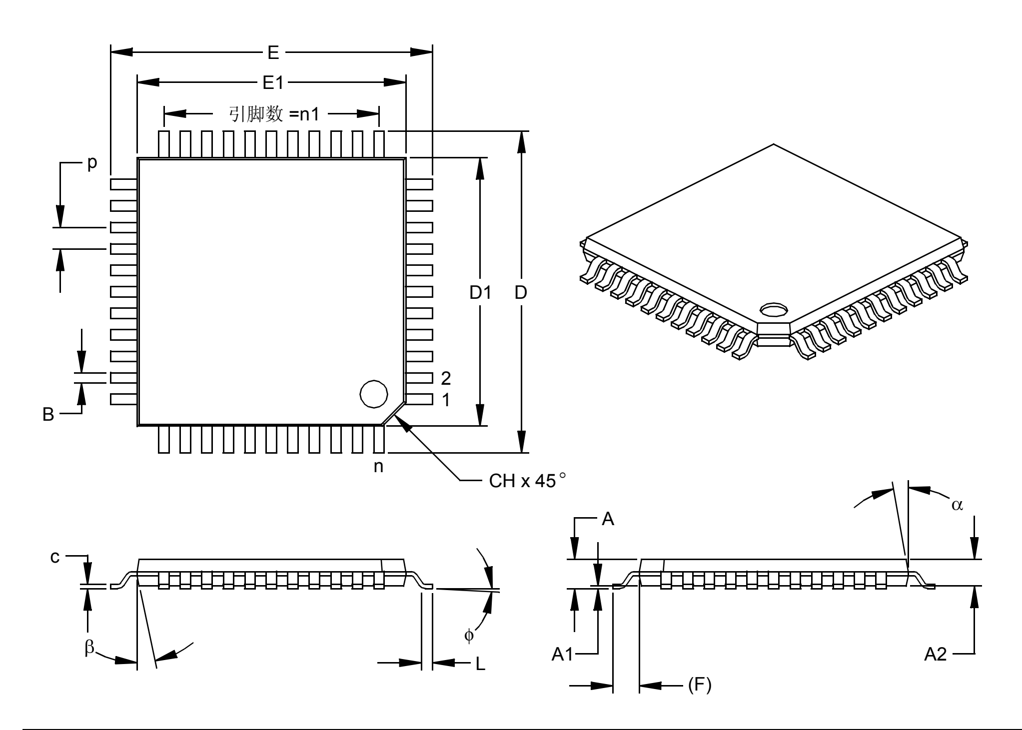
40 引脚塑料双列直插封装 （P）— 600 mil （PDIP）



单位		英寸 *			毫米		
尺寸范围		最小值	标准值	最大值	最小值	标准值	最大值
引脚数	n		40			40	
引脚间距	p		.100			2.54	
顶端到固定面高度	A	.160	.175	.190	4.06	4.45	4.83
塑模封装厚度	A2	.140	.150	.160	3.56	3.81	4.06
塑模底端到固定面高度	A1	.015			0.38		
肩到肩宽度	E	.595	.600	.625	15.11	15.24	15.88
塑模封装宽度	E1	.530	.545	.560	13.46	13.84	14.22
总长度	D	2.045	2.058	2.065	51.94	52.26	52.45
引脚尖到固定面高度	L	.120	.130	.135	3.05	3.30	3.43
引脚厚度	c	.008	.012	.015	0.20	0.29	0.38
引脚上部宽度	B1	.030	.050	.070	0.76	1.27	1.78
引脚下部宽度	B	.014	.018	.022	0.36	0.46	0.56
总排列间距 §	eB	.620	.650	.680	15.75	16.51	17.27
塑模顶部锥度	α	5	10	15	5	10	15
塑模底部锥度	β	5	10	15	5	10	15

* 控制参数
§ 重要特性参数
注：
尺寸 D 和 E1 不包括塑模毛边或突起。每侧的塑模毛边或突起不得超过 0.010 英寸（0.254 毫米）。
等同 JEDEC 规范：MO-011
图号 C04-016

44 引脚塑料薄型四列扁平封装 （PT）主体 10x10x1 mm， 1.0/0.10 mm 引脚形式 （TQFP）

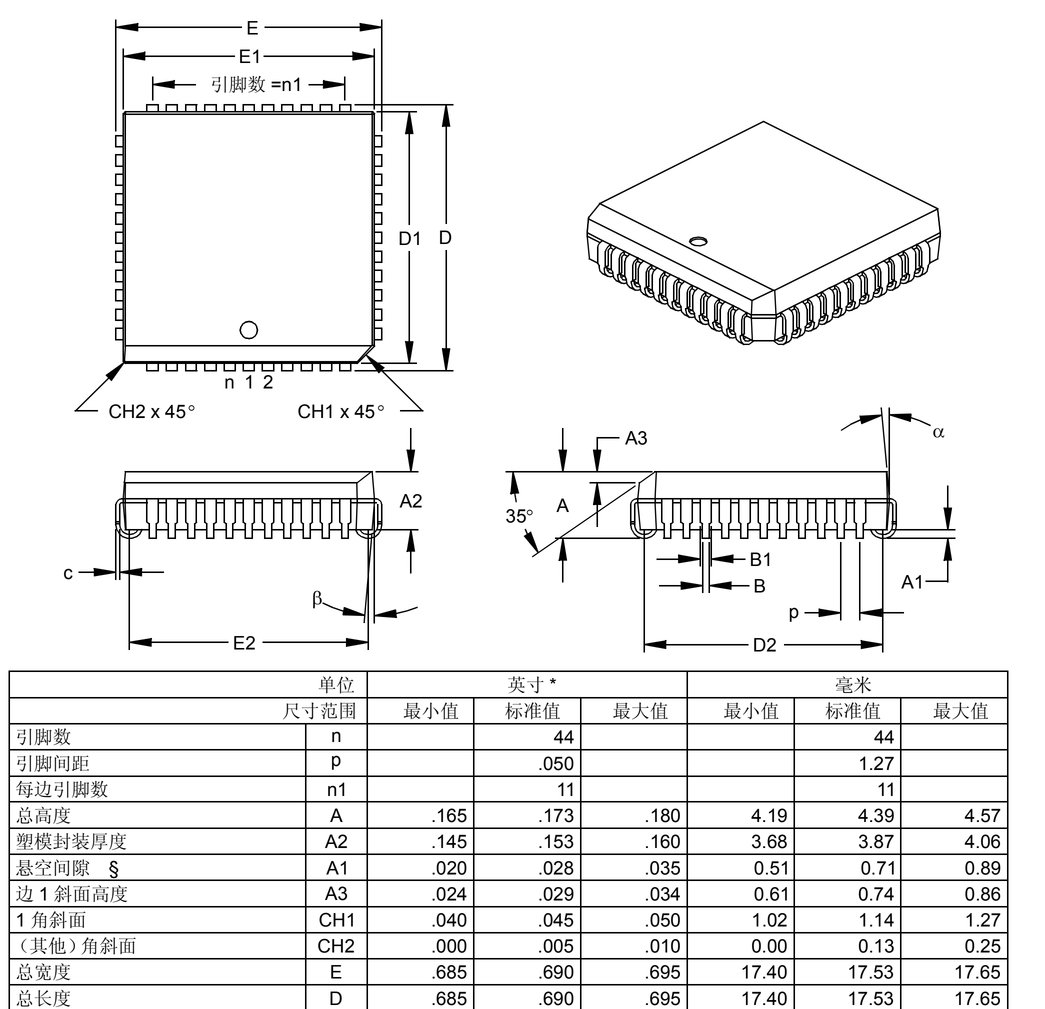


单位		英寸			毫米 *		
尺寸范围		最小值	标准值	最大值	最小值	标准值	最大值
引脚数	n		44			44	
引脚间距	p		.031			0.80	
每边引脚数	n1		11			11	
总高度	A	.039	.043	.047	1.00	1.10	1.20
塑模封装厚度	A2	.037	.039	.041	0.95	1.00	1.05
悬空间隙 §	A1	.002	.004	.006	0.05	0.10	0.15
底脚长度	L	.018	.024	.030	0.45	0.60	0.75
占用面积（参考）	(F)		.039		1.00		
底脚倾斜角	φ	0	3.5	7	0	3.5	7
总宽度	E	.463	.472	.482	11.75	12.00	12.25
总长度	D	.463	.472	.482	11.75	12.00	12.25
塑模封装宽度	E1	.390	.394	.398	9.90	10.00	10.10
塑模封装长度	D1	.390	.394	.398	9.90	10.00	10.10
引脚厚度	c	.004	.006	.008	0.09	0.15	0.20
引脚宽度	B	.012	.015	.017	0.30	0.38	0.44
引脚 1 角斜面	CH	.025	.035	.045	0.64	0.89	1.14
塑模顶部锥度	α	5	10	15	5	10	15
塑模底部锥度	β	5	10	15	5	10	15

* 控制参数
§ 重要特性参数

注:
尺寸 D 和 E1 不包括塑模毛边或突起。每侧的塑模毛边或突起不得超过 0.010 英寸（0.254 毫米）。
等同 JEDEC 规范: MS-026
图号 C04-076

44 引脚塑封有引线芯片载体（L）—方形（PLCC）



单位		英寸 *			毫米		
尺寸范围		最小值	标准值	最大值	最小值	标准值	最大值
引脚数	n		44			44	
引脚间距	p		.050			1.27	
每边引脚数	n1		11			11	
总高度	A	.165	.173	.180	4.19	4.39	4.57
塑模封装厚度	A2	.145	.153	.160	3.68	3.87	4.06
悬空间隙 §	A1	.020	.028	.035	0.51	0.71	0.89
边 1 斜面高度	A3	.024	.029	.034	0.61	0.74	0.86
1 角斜面	CH1	.040	.045	.050	1.02	1.14	1.27
（其他）角斜面	CH2	.000	.005	.010	0.00	0.13	0.25
总宽度	E	.685	.690	.695	17.40	17.53	17.65
总长度	D	.685	.690	.695	17.40	17.53	17.65
塑模封装宽度	E1	.650	.653	.656	16.51	16.59	16.66
塑模封装长度	D1	.650	.653	.656	16.51	16.59	16.66
占用面积宽度	E2	.590	.620	.630	14.99	15.75	16.00
占用面积长度	D2	.590	.620	.630	14.99	15.75	16.00
引脚厚度	c	.008	.011	.013	0.20	0.27	0.33
引脚上部宽度	B1	.026	.029	.032	0.66	0.74	0.81
引脚下部宽度	B	.013	.020	.021	0.33	0.51	0.53
塑模顶部锥度	α	0	5	10	0	5	10
塑模底部锥度	β	0	5	10	0	5	10

* 控制参数
§ 重要特性参数
注：
尺寸 D 和 E1 不包括塑模毛边或突起。每侧的塑模毛边或突起不得超过 0.010 英寸（0.254 毫米）。
等同 JEDEC 规范：MO-047
图号 C04-048

注：

附录 A： 版本历史

版本 A （2001 年 6 月）

PIC18FXX2 系列数据手册的最初版本。

版本 B （2002 年 8 月）

该版本包含了 DC 和 AC 特性图表。已更新第 22.0 节中的电气规范，并对数据手册正文稍作修正。

附录 B： 器件差异

表 B-1 为本数据手册中所列器件的差异。

表 B-1： 器件差异

功能特性	PIC18F242	PIC18F252	PIC18F442	PIC18F452
程序存储器 （KB）	16	32	16	32
数据存储器 （字节）	768	1536	768	1536
A/D 通道数	5	5	8	8
并行从动端口 （PSP)	无	无	有	有
封装类型	28 引脚 DIP 28 引脚 SOIC	28 引脚 DIP 28 引脚 SOIC	40 引脚 DIP 44 引脚 PLCC 44 引脚 TQFP	40 引脚 DIP 44 引脚 PLCC 44 引脚 TQFP

附录 C： 转换注意事项

本附录讨论了器件的老版本升级至此数据手册中所列版本时需要注意的事项。这些变化通常是因采用的加工工艺技术方面的差别所引起的。从 PIC16C74A 至 PIC16C74B 的变化就是这类转换的一个例子。

不适用

附录 D： 从基本器件迁移到增强型器件

本节讨论如何从基本型器件（如 PIC16C5X）迁移到增强型 MCU 器件 （如 PIC18FXXX）。

下表列出了对 PIC16C5X 单片机系列所做的修改：

当前不可用

附录 E： 从中档器件迁移到增强型器件

在 AN716“Migrating Designs from PIC16C74A/74B to PIC18F442”中详细讨论了中档 MCU 器件（如 PIC16CXXX）与增强型器件（如 PIC18FXXX）之间的区别。虽然所讨论的变化都是针对特定器件的，但是通常可以适用于中档器件至增强器件的所有迁移。

此应用笔记的文献编号为 DS00716。

附录 F： 从高档器件迁移到增强型器件

在 AN726“PIC17CXXX to PIC18FXXX Migration”中详细讨论了高档 MCU 器件（如 PIC17CXXX）迁移到增强型器件（如 PIC18FXXX）的步骤及两者间的区别。此应用笔记的文献编号为 DS00726。

注：

索引

数字

C

D

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

W

X

Y

Z

注：

MICROCHIP 网站

Microchip 网站（www.microchip.com）为客户提供在线支持。客户可通过该网站方便地获取文件和信息。只要使用常用的因特网浏览器即可访问。网站提供以下信息：

- **产品支持**——数据手册和勘误表、应用笔记和样本程序、设计资源、用户指南以及硬件支持文档、最新的软件版本以及存档软件
- **一般技术支持**——常见问题（FAQ）、技术支持请求、在线讨论组以及 Microchip 顾问计划成员名单
- **Microchip 业务**——产品选型和订购指南、最新 Microchip 新闻稿、研讨会和活动安排表、Microchip 销售办事处、代理商以及工厂代表列表

变更通知客户服务

Microchip 的变更通知客户服务有助于客户了解 Microchip 产品的最新信息。注册客户可在他们感兴趣的某个产品系列或开发工具发生变更、更新、发布新版本或勘误表时，收到电子邮件通知。

客户支持

Microchip 产品的用户可通过以下渠道获得帮助：

- 代理商或代表
- 当地销售办事处
- 应用工程师（FAE）
- 技术支持
- 开发系统信息热线

客户应联系其代理商、代表或应用工程师（FAE）寻求支持。当地销售办事处也可为客户提供帮助。本文档后附有销售办事处的联系方式。

也可通过 http：//support.microchip.com 获得网上技术支持

此外，我们还设有一条开发系统信息热线，列出了 Microchip 开发系统软件产品的最新版本。此热线还向客户提供如何取得当前可用的升级软件包的信息。

开发系统信息热线号码为：

1-800-755-2345 ——美国和加拿大大部分地区

800-820-6247——中国免费技术咨询热线

1-480-792-7302——其他国家或地区。

读者反馈表

我们努力为您提供最佳文档，以确保您能够成功使用 Microchip 产品。如果您对文档的组织、条理性、主题及其他有助于提高文档质量的方面有任何意见或建议，请填写本反馈表并传真给我公司 TRC 经理，传真号码为 86-21-5407-5066。

请填写以下信息，并从下面各方面提出您对本文档的意见。

致: TRC 经理　　　　总页数 ________

关于: 读者反馈

发自: 姓名 ________________________

公司 ________________________

地址 ________________________

国家 / 省份 / 城市 / 邮编 ________________________

电话: (______) ________________　　传真: (______) ________________

应用（选填）:

您希望收到回复吗？是____ 否____

器件: PIC18FXX2　　文献编号: DS39564B_CN

问题:

1. 本文档中哪些部分最有特色？

2. 本文档是否满足了您的软硬件开发要求？如何满足的？

3. 您认为本文档的组织结构便于理解吗？如果不便于理解，那么问题何在？

4. 您认为本文档应该添加哪些内容以改善其结构和主题？

5. 您认为本文档中可以删减哪些内容，而又不会影响整体使用效果？

6. 本文档中是否存在错误或误导信息？如果存在，请指出是什么信息及其具体页数。

7. 您认为本文档还有哪些方面有待改进？

PIC18FXX2 产品标识体系

欲订货，或获取价格、交货等信息，请与我公司生产厂或各销售办事处联系。

器件编号	–	X	/XX	XXX
器件		温度范围	封装	模式

器件	PIC18FXX2(1) 和 PIC18FXX2T(2) VDD 范围为 4.2V 至 5.5V PIC18LFXX2(1) 和 PIC18LFXX2T(2) VDD 范围为 2.5V 至 5.5V
温度范围	I = -40°C 至 +85°C（工业级） E = -40°C 至 +125°C（扩展级）
封装	PT = TQFP （薄型四方扁平封装） SO = SOIC SP = 塑料窄条 DIP P = PDIP L = PLCC
模式	QTP、 SQTP、编码或特殊要求 （空白为其他情况）

示例：

a) PIC18LF452 - I/P 301 = 工业级温度，PDIP 封装，扩展级 VDD 范围，QTP 模式 #301。
b) PIC18LF242 - I/SO = 工业级温度，SOIC 封装，扩展级 VDD 范围。
c) PIC18F442 - E/P = 扩展级温度，PDIP 封装，一般 VDD 范围。

注 **1:** F = 标准电压范围
LF = 宽电压范围
2: T = 卷带式（仅 SOIC、 PLCC 和 TQFP 封装）

销售与技术支持

数据手册

现有器件可能带有一份勘误表，描述了实际运行与数据手册中记载内容之间存在的细微差异以及建议的变通方法。一旦我们了解到器件 / 文档存在某些差异时，就会发布勘误表。勘误表上将注明其所适用的硅片版本和文件版本。

欲了解某一器件是否存在勘误表，请通过以下方式之一查询：

- Microchip 网站 http://www.microchip.com
- 当地 Microchip 销售办事处 （见最后一页）

在联络销售办事处时，请说明您所使用的器件型号、硅片版本和数据手册版本 （包括文献编号）。

最新信息客户通知系统

欲及时获知 Microchip 产品的最新信息，请到我公司网站 **www.microchip.com** 上注册。



全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://support.microchip.com
网址：www.microchip.com

亚特兰大 **Atlanta**
Alpharetta, GA
Tel: 1-770-640-0034
Fax: 1-770-640-0307

波士顿 **Boston**
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 **Chicago**
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

达拉斯 **Dallas**
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 **Detroit**
Farmington Hills, MI
Tel: 1-248-538-2250
Fax: 1-248-538-2260

科科莫 **Kokomo**
Kokomo, IN
Tel: 1-765-864-8360
Fax: 1-765-864-8387

洛杉矶 **Los Angeles**
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608

圣何塞 **San Jose**
Mountain View, CA
Tel: 1-650-215-1444
Fax: 1-650-961-0286

加拿大多伦多 **Toronto**
Mississauga, Ontario,
Canada
Tel: 1-905-673-0699
Fax: 1-905-673-6509

亚太地区

中国 − 北京
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

中国 − 成都
Tel: 86-28-8676-6200
Fax: 86-28-8676-6599

中国 − 福州
Tel: 86-591-8750-3506
Fax: 86-591-8750-3521

中国 − 香港特别行政区
Tel: 852-2401-1200
Fax: 852-2401-3431

中国 − 上海
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

中国 − 沈阳
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

中国 − 深圳
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

中国 − 顺德
Tel: 86-757-2839-5507
Fax: 86-757-2839-5571

中国 − 青岛
Tel: 86-532-502-7355
Fax: 86-532-502-7205

台湾地区 − 高雄
Tel: 886-7-536-4818
Fax: 886-7-536-4803

台湾地区 − 台北
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

台湾地区 − 新竹
Tel: 886-3-572-9526
Fax: 886-3-572-6459

亚太地区

澳大利亚 **Australia - Sydney**
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

印度 **India - Bangalore**
Tel: 91-80-2229-0061
Fax: 91-80-2229-0062

印度 **India - New Delhi**
Tel: 91-11-5160-8631
Fax: 91-11-5160-8632

日本 **Japan - Kanagawa**
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

韩国 **Korea - Seoul**
Tel: 82-2-554-7200
Fax: 82-2-558-5932 或
82-2-558-5934

新加坡 **Singapore**
Tel: 65-6334-8870
Fax: 65-6334-8850

欧洲

奥地利 **Austria - Weis**
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

丹麦 **Denmark - Ballerup**
Tel: 45-4450-2828
Fax: 45-4485-2829

法国 **France - Massy**
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 **Germany - Ismaning**
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

意大利 **Italy - Milan**
Tel: 39-0331-742611
Fax: 39-0331-466781

荷兰 **Netherlands - Drunen**
Tel: 31-416-690399
Fax: 31-416-690340

英国 **England - Berkshire**
Tel: 44-118-921-5869
Fax: 44-118-921-5820

03/01/05